



AMD 64-Bit Technology

AMD x86-64 Architecture

Programmer's Manual

Volume 4:

128-Bit Media Instructions

Publication No.	Revision	Date
26568	3.00	August 2002

© 2002 Advanced Micro Devices, Inc. All rights reserved.

The contents of this document are provided in connection with Advanced Micro Devices, Inc. ("AMD") products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. Except as set forth in AMD's Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

Trademarks

AMD, the AMD logo, AMD Athlon, AMD Duron, AMD-K5, 3DNow!, and combinations thereof, are trademarks and Am486, Am5_x86 and AMD-K6 are a registered trademarks of Advanced Micro Devices, Inc.

MMX is a trademark and Pentium is a registered trademark of Intel Corporation.

Windows NT is a registered trademark of Microsoft Corp.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Contents

Figures	vii
Tables	ix
Preface	xi
About This Book	xi
Audience	xi
Organization	xi
Definitions	xii
Related Documents	xxiii
1 128-Bit Media Instruction Reference	1
ADDPD	4
ADDPS	6
ADDSD	8
ADDSS	11
ANDNPD	14
ANDNPS	16
ANDPD	18
ANDPS	20
CMPPD	22
CMPPS	26
CMPSD	29
CMPSS	32
COMISD	35
COMISS	38
CVTDQ2PD	41
CVTDQ2PS	43
CVTPD2DQ	45
CVTPD2PI	47
CVTPD2PS	50
CVTPI2PD	53
CVTPI2PS	55
CVTPS2DQ	58
CVTPS2PD	60
CVTPS2PI	62
CVTSD2SI	65
CVTSD2SS	68
CVTSI2SD	71
CVTSI2SS	74
CVTSS2SD	77
CVTSS2SI	80
CVTTPD2DQ	83
CVTTPD2PI	85

CVTTTPS2DQ	88
CVTTTPS2PI	90
CVTTSD2SI	93
CVTTSS2SI	96
DIVPD	99
DIVPS	102
DIVSD	105
DIVSS	108
FXRSTOR	111
FXSAVE	113
LDMXCSR	115
MASKMOVDQU	117
MAXPD	119
MAXPS	121
MAXSD	124
MAXSS	127
MINPD	130
MINPS	132
MINSD	135
MINSS	138
MOVAPD	141
MOVAPS	143
MOVD	146
MOVDQ2Q	149
MOVDQA	151
MOVDQU	153
MOVHLPS	155
MOVHPD	157
MOVHPS	159
MOVLHPS	161
MOVLPD	163
MOVLPS	165
MOVMSKPD	167
MOVMSKPS	169
MOVNTDQ	171
MOVNTPD	173
MOVNTPS	175
MOVQ	177
MOVQ2DQ	179
MOVSD	181
MOVSS	184
MOVUPD	187
MOVUPS	189
MULPD	191
MULPS	194
MULSD	197
MULSS	200

ORPD	203
ORPS	205
PACKSSDW	207
PACKSSWB	209
PACKUSWB	211
PADDB	213
PADD	215
PADDQ	217
PADDSB	219
PADDSW	221
PADDUSB	223
PADDUSW	225
PADDW	227
PAND	229
PANDN	231
PAVGB	233
PAVGW	235
PCMPEQB	237
PCMPEQD	239
PCMPEQW	241
PCMPGTB	243
PCMPGTD	245
PCMPGTW	247
PEXTRW	249
PINSRW	251
PMADDWD	254
PMAXSW	256
PMAXUB	258
PMINSW	260
PMINUB	262
PMOVMSKB	264
PMULHUW	266
PMULHW	268
PMULLW	270
PMULUDQ	272
POR	274
PSADBW	276
PSHUFD	278
PSHUFHW	281
PSHUFLW	284
PSLLD	287
PSLLDQ	289
PSLLQ	291
PSLLW	293
PSRAD	295
PSRAW	298
PSRLD	301

PSRLDQ	303
PSRLQ	305
PSRLW	307
PSUBB	310
PSUBD	312
PSUBQ	314
PSUBSB	316
PSUBSW	318
PSUBUSB	320
PSUBUSW	322
PSUBW	324
PUNPCKHBW	326
PUNPCKHDQ	328
PUNPCKHQDQ	330
PUNPCKHWD	332
PUNPCKLBW	334
PUNPCKLDQ	336
PUNPCKLQDQ	338
PUNPCKLWD	340
PXOR	342
RCPPS	344
RCPSS	346
RSQRTPS	348
RSQRTSS	350
SHUFPD	352
SHUFPS	355
SQRTPD	358
SQRTPS	360
SQRTSD	363
SQRTSS	365
STMXCSR	367
SUBPD	369
SUBPS	372
SUBSD	375
SUBSS	378
UCOMISD	381
UCOMISS	384
UNPCKHPD	387
UNPCKHPS	389
UNPCKLPD	391
UNPCKLPS	393
XORPD	395
XORPS	397
Index	399

Figures

Figure 1-1. Diagram Conventions for 128-Bit Media Instructions	2
--	---

Tables

Table 1-1.	Immediate Operand Values for Compare Operations	23
Table 1-2.	Immediate-Byte Operand Encoding for 128-Bit PEXTRW . . .	249
Table 1-3.	Immediate-Byte Operand Encoding for 128-Bit PINSRW . . .	252
Table 1-4.	Immediate-Byte Operand Encoding for PSHUFD	279
Table 1-5.	Immediate-Byte Operand Encoding for PSHUFHW	282
Table 1-6.	Immediate-Byte Operand Encoding for PSHUFLW	285
Table 1-7.	Immediate-Byte Operand Encoding for SHUFPD	353
Table 1-8.	Immediate-Byte Operand Encoding for SHUFPS	356

Preface

About This Book

This book is part of a multivolume work entitled the *AMD x86-64 Architecture Programmer's Manual*. This table lists each volume and its order number.

Title	Order No.
Volume 1, <i>Application Programming</i>	24592
Volume 2, <i>System Programming</i>	24593
Volume 3, <i>General-Purpose and System Instructions</i>	24594
Volume 4, <i>128-Bit Media Instructions</i>	26568
Volume 5, <i>64-Bit Media and x87 Floating-Point Instructions</i>	26569

Audience

This volume (Volume 4) is intended for all programmers writing application or system software for processors that implement the x86-64 architecture.

Organization

Volumes 3, 4, and 5 describe the x86-64 architecture's instruction set in detail. Together, they cover each instruction's mnemonic syntax, opcodes, functions, affected flags, and possible exceptions.

The x86-64 instruction set is divided into five subsets:

- General-purpose instructions
- System instructions
- 128-bit media instructions
- 64-bit media instructions
- x87 floating-point instructions

Several instructions belong to—and are described identically in—multiple instruction subsets.

This volume describes the 128-bit media instructions. The index at the end cross-references topics within this volume. For other topics relating to the x86-64 architecture, and for information on instructions in other subsets, see the tables of contents and indexes of the other volumes.

Definitions

Many of the following definitions assume an in-depth knowledge of the legacy x86 architecture. See “Related Documents” on page xxiii for descriptions of the legacy x86 architecture.

Terms and Notation

In addition to the notation described below, “Opcode-Syntax Notation” in Volume 3 describes notation relating specifically to opcodes.

1011b

A binary value—in this example, a 4-bit value.

F0EAh

A hexadecimal value—in this example a 2-byte value.

[1,2)

A range that includes the left-most value (in this case, 1) but excludes the right-most value (in this case, 2).

7–4

A bit range, from bit 7 to 4, inclusive. The high-order bit is shown first.

128-bit media instructions

Instructions that use the 128-bit XMM registers. These are a combination of the SSE and SSE2 instruction sets.

64-bit media instructions

Instructions that use the 64-bit MMX™ registers. These are primarily a combination of MMX and 3DNow!™ instruction sets, with some additional instructions from the SSE and SSE2 instruction sets.

16-bit mode

Legacy mode or compatibility mode in which a 16-bit address size is active. See *legacy mode* and *compatibility mode*.

32-bit mode

Legacy mode or compatibility mode in which a 32-bit address size is active. See *legacy mode* and *compatibility mode*.

64-bit mode

A submode of *long mode*. In 64-bit mode, the default address size is 64 bits and new features, such as register extensions, are supported for system and application software.

#GP(0)

Notation indicating a general-protection exception (#GP) with error code of 0.

absolute

Said of a displacement that references the base of a code segment rather than an instruction pointer. Contrast with *relative*.

biased exponent

The sum of a floating-point value's exponent and a constant bias for a particular floating-point data type. The bias makes the range of the biased exponent always positive, which allows reciprocation without overflow.

byte

Eight bits.

clear

To write a bit value of 0. Compare *set*.

compatibility mode

A submode of *long mode*. In compatibility mode, the default address size is 32 bits, and legacy 16-bit and 32-bit applications run without modification.

commit

To irreversibly write, in program order, an instruction's result to software-visible storage, such as a register (including flags), the data cache, an internal write buffer, or memory.

CPL

Current privilege level.

CR0–CR4

A register range, from register CR0 through CR4, inclusive, with the low-order register first.

CR0.PE = 1

Notation indicating that the PE bit of the CR0 register has a value of 1.

direct

Referencing a memory location whose address is included in the instruction's syntax as an immediate operand. The address may be an absolute or relative address. Compare *indirect*.

dirty data

Data held in the processor's caches or internal buffers that is more recent than the copy held in main memory.

displacement

A signed value that is added to the base of a segment (absolute addressing) or an instruction pointer (relative addressing). Same as *offset*.

doubleword

Two words, or four bytes, or 32 bits.

double quadword

Eight words, or 16 bytes, or 128 bits. Also called *octword*.

DS:rSI

The contents of a memory location whose segment address is in the DS register and whose offset relative to that segment is in the rSI register.

EFER.LME = 0

Notation indicating that the LME bit of the EFER register has a value of 0.

effective address size

The address size for the current instruction after accounting for the default address size and any address-size override prefix.

effective operand size

The operand size for the current instruction after accounting for the default operand size and any operand-size override prefix.

element

See *vector*.

exception

An abnormal condition that occurs as the result of executing an instruction. The processor's response to an exception depends on the type of the exception. For all exceptions except 128-bit media SIMD floating-point exceptions and x87 floating-point exceptions, control is transferred to the handler (or service routine) for that exception, as defined by the exception's vector. For floating-point exceptions defined by the IEEE 754 standard, there are both masked and unmasked responses. When unmasked, the exception handler is called, and when masked, a default response is provided instead of calling the handler.

FF /0

Notation indicating that FF is the first byte of an opcode, and a subfield in the second byte has a value of 0.

flush

An often ambiguous term meaning (1) writeback, if modified, and invalidate, as in “flush the cache line,” or (2) invalidate, as in “flush the pipeline,” or (3) change a value, as in “flush to zero.”

GDT

Global descriptor table.

IDT

Interrupt descriptor table.

IGN

Ignore. Field is ignored.

indirect

Referencing a memory location whose address is in a register or other memory location. The address may be an absolute or relative address. Compare *direct*.

IRB

The virtual-8086 mode interrupt-redirection bitmap.

IST

The long-mode interrupt-stack table.

IVT

The real-address mode interrupt-vector table.

LDT

Local descriptor table.

legacy x86

The legacy x86 architecture. See “Related Documents” on page xxiii for descriptions of the legacy x86 architecture.

legacy mode

An operating mode of the x86-64 architecture in which existing 16-bit and 32-bit applications and operating systems run without modification. A processor implementation of the x86-64 architecture can run in either *long mode* or *legacy mode*. Legacy mode has three submodes, *real mode*, *protected mode*, and *virtual-8086 mode*.

long mode

An operating mode unique to the x86-64 architecture. A processor implementation of the x86-64 architecture can run in either *long mode* or *legacy mode*. Long mode has two submodes, *64-bit mode* and *compatibility mode*.

lsb

Least-significant bit.

LSB

Least-significant byte.

main memory

Physical memory, such as RAM and ROM (but not cache memory) that is installed in a particular computer system.

mask

(1) A control bit that prevents the occurrence of a floating-point exception from invoking an exception-handling routine. (2) A field of bits used for a control purpose.

MBZ

Must be zero. If software attempts to set an MBZ bit to 1, a general-protection exception (#GP) occurs.

memory

Unless otherwise specified, *main memory*.

ModRM

A byte following an instruction opcode that specifies address calculation based on mode (Mod), register (R), and memory (M) variables.

moffset

A direct memory offset. In other words, a displacement that is added to the base of a code segment (for absolute addressing) or to an instruction pointer (for addressing relative to the instruction pointer, as in RIP-relative addressing).

msb

Most-significant bit.

MSB

Most-significant byte.

multimedia instructions

A combination of *128-bit media instructions* and *64-bit media instructions*.

octword

Same as *double quadword*.

offset

Same as *displacement*.

overflow

The condition in which a floating-point number is larger in magnitude than the largest, finite, positive or negative number that can be represented in the data-type format being used.

packed

See *vector*.

PAE

Physical-address extensions.

physical memory

Actual memory, consisting of *main memory* and cache.

probe

A check for an address in a processor's caches or internal buffers. *External probes* originate outside the processor, and *internal probes* originate within the processor.

protected mode

A submode of *legacy mode*.

quadword

Four words, or eight bytes, or 64 bits.

RAZ

Read as zero (0), regardless of what is written.

real-address mode

See *real mode*.

real mode

A short name for *real-address mode*, a submode of *legacy mode*.

relative

Referencing with a displacement (also called offset) from an instruction pointer rather than the base of a code segment. Contrast with *absolute*.

REX

An instruction prefix that specifies a 64-bit operand size and provides access to additional registers.

RIP-relative addressing

Addressing relative to the 64-bit RIP instruction pointer. Compare *moffset*.

set

To write a bit value of 1. Compare *clear*.

SIB

A byte following an instruction opcode that specifies address calculation based on scale (S), index (I), and base (B).

SIMD

Single instruction, multiple data. See *vector*.

SSE

Streaming SIMD extensions instruction set. See *128-bit media instructions* and *64-bit media instructions*.

SSE2

Extensions to the SSE instruction set. See *128-bit media instructions* and *64-bit media instructions*.

sticky bit

A bit that is set or cleared by hardware and that remains in that state until explicitly changed by software.

TOP

The x87 top-of-stack pointer.

TPR

Task-priority register (CR8).

TSS

Task-state segment.

underflow

The condition in which a floating-point number is smaller in magnitude than the smallest nonzero, positive or negative number that can be represented in the data-type format being used.

vector

(1) A set of integer or floating-point values, called *elements*, that are packed into a single operand. Most of the 128-bit and 64-bit media instructions use vectors as operands. Vectors are also called *packed* or *SIMD* (single-instruction multiple-data) operands.

(2) An index into an interrupt descriptor table (IDT), used to access exception handlers. Compare *exception*.

virtual-8086 mode

A submode of *legacy mode*.

word

Two bytes, or 16 bits.

x86

See *legacy x86*.

Registers

In the following list of registers, the names are used to refer either to a given register or to the contents of that register:

AH–DH

The high 8-bit AH, BH, CH, and DH registers. Compare *AL–DL*.

AL–DL

The low 8-bit AL, BL, CL, and DL registers. Compare *AH–DH*.

AL–r15B

The low 8-bit AL, BL, CL, DL, SIL, DIL, BPL, SPL, and R8B–R15B registers, available in 64-bit mode.

BP

Base pointer register.

CR_n

Control register number *n*.

CS

Code segment register.

eAX–eSP

The 16-bit AX, BX, CX, DX, DI, SI, BP, and SP registers or the 32-bit EAX, EBX, ECX, EDX, EDI, ESI, EBP, and ESP registers. Compare *rAX–rSP*.

EBP

Extended base pointer register.

EFER

Extended features enable register.

eFLAGS

16-bit or 32-bit flags register. Compare *rFLAGS*.

EFLAGS

32-bit (extended) flags register.

eIP

16-bit or 32-bit instruction-pointer register. Compare *rIP*.

EIP

32-bit (extended) instruction-pointer register.

FLAGS

16-bit flags register.

GDTR

Global descriptor table register.

GPRs

General-purpose registers. For the 16-bit data size, these are AX, BX, CX, DX, DI, SI, BP, and SP. For the 32-bit data size, these are EAX, EBX, ECX, EDX, EDI, ESI, EBP, and ESP. For the 64-bit data size, these include RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, and R8–R15.

IDTR

Interrupt descriptor table register.

IP

16-bit instruction-pointer register.

LDTR

Local descriptor table register.

MSR

Model-specific register.

r8–r15

The 8-bit R8B–R15B registers, or the 16-bit R8W–R15W registers, or the 32-bit R8D–R15D registers, or the 64-bit R8–R15 registers.

rAX–rSP

The 16-bit AX, BX, CX, DX, DI, SI, BP, and SP registers, or the 32-bit EAX, EBX, ECX, EDX, EDI, ESI, EBP, and ESP registers, or the 64-bit RAX, RBX, RCX, RDX, RDI, RSI, RBP, and RSP registers. Replace the placeholder *r* with

nothing for 16-bit size, “E” for 32-bit size, or “R” for 64-bit size.

RAX

64-bit version of the EAX register.

RBP

64-bit version of the EBP register.

RBX

64-bit version of the EBX register.

RCX

64-bit version of the ECX register.

RDI

64-bit version of the EDI register.

RDX

64-bit version of the EDX register.

rFLAGS

16-bit, 32-bit, or 64-bit flags register. Compare *RFLAGS*.

RFLAGS

64-bit flags register. Compare *rFLAGS*.

rIP

16-bit, 32-bit, or 64-bit instruction-pointer register. Compare *RIP*.

RIP

64-bit instruction-pointer register.

RSI

64-bit version of the ESI register.

RSP

64-bit version of the ESP register.

SP

Stack pointer register.

SS

Stack segment register.

TPR

Task priority register, a new register introduced in the x86-64 architecture to speed interrupt management.

TR

Task register.

Endian Order

The x86 and x86-64 architectures address memory using little-endian byte-ordering. Multibyte values are stored with their least-significant byte at the lowest byte address, and they are illustrated with their least significant byte at the right side. Strings are illustrated in reverse order, because the addresses of their bytes increase from right to left.

Related Documents

- Peter Abel, *IBM PC Assembly Language and Programming*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
- Rakesh Agarwal, *80x86 Architecture & Programming: Volume II*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
- AMD, *AMD-K6™ MMX™ Enhanced Processor Multimedia Technology*, Sunnyvale, CA, 2000.
- AMD, *3DNow!™ Technology Manual*, Sunnyvale, CA, 2000.
- AMD, *AMD Extensions to the 3DNow!™ and MMX™ Instruction Sets*, Sunnyvale, CA, 2000.
- Don Anderson and Tom Shanley, *Pentium Processor System Architecture*, Addison-Wesley, New York, 1995.
- Nabajyoti Barkakati and Randall Hyde, *Microsoft Macro Assembler Bible*, Sams, Carmel, Indiana, 1992.
- Barry B. Brey, *8086/8088, 80286, 80386, and 80486 Assembly Language Programming*, Macmillan Publishing Co., New York, 1994.
- Barry B. Brey, *Programming the 80286, 80386, 80486, and Pentium Based Personal Computer*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
- Ralf Brown and Jim Kyle, *PC Interrupts*, Addison-Wesley, New York, 1994.
- Penn Brumm and Don Brumm, *80386/80486 Assembly Language Programming*, Windcrest McGraw-Hill, 1993.
- Geoff Chappell, *DOS Internals*, Addison-Wesley, New York, 1994.

- Chips and Technologies, Inc. *Super386 DX Programmer's Reference Manual*, Chips and Technologies, Inc., San Jose, 1992.
- John Crawford and Patrick Gelsinger, *Programming the 80386*, Sybex, San Francisco, 1987.
- Cyrix Corporation, *5x86 Processor BIOS Writer's Guide*, Cyrix Corporation, Richardson, TX, 1995.
- Cyrix Corporation, *M1 Processor Data Book*, Cyrix Corporation, Richardson, TX, 1996.
- Cyrix Corporation, *MX Processor MMX Extension Opcode Table*, Cyrix Corporation, Richardson, TX, 1996.
- Cyrix Corporation, *MX Processor Data Book*, Cyrix Corporation, Richardson, TX, 1997.
- Jeffrey P. Doyer, *Introduction to Protected Mode Programming*, course materials for an onsite class, 1992.
- Ray Duncan, *Extending DOS: A Programmer's Guide to Protected-Mode DOS*, Addison Wesley, NY, 1991.
- William B. Giles, *Assembly Language Programming for the Intel 80xxx Family*, Macmillan, New York, 1991.
- Frank van Gilluwe, *The Undocumented PC*, Addison-Wesley, New York, 1994.
- John L. Hennessy and David A. Patterson, *Computer Architecture*, Morgan Kaufmann Publishers, San Mateo, CA, 1996.
- Thom Hogan, *The Programmer's PC Sourcebook*, Microsoft Press, Redmond, WA, 1991.
- Hal Katircioglu, *Inside the 486, Pentium, and Pentium Pro*, Peer-to-Peer Communications, Menlo Park, CA, 1997.
- IBM Corporation, *486SLC Microprocessor Data Sheet*, IBM Corporation, Essex Junction, VT, 1993.
- IBM Corporation, *486SLC2 Microprocessor Data Sheet*, IBM Corporation, Essex Junction, VT, 1993.
- IBM Corporation, *80486DX2 Processor Floating Point Instructions*, IBM Corporation, Essex Junction, VT, 1995.
- IBM Corporation, *80486DX2 Processor BIOS Writer's Guide*, IBM Corporation, Essex Junction, VT, 1995.
- IBM Corporation, *Blue Lightning 486DX2 Data Book*, IBM Corporation, Essex Junction, VT, 1994.

- Institute of Electrical and Electronics Engineers, *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE Std 754-1985.
- Institute of Electrical and Electronics Engineers, *IEEE Standard for Radix-Independent Floating-Point Arithmetic*, ANSI/IEEE Std 854-1987.
- Muhammad Ali Mazidi and Janice Gillispie Mazidi, *80X86 IBM PC and Compatible Computers*, Prentice-Hall, Englewood Cliffs, NJ, 1997.
- Hans-Peter Messmer, *The Indispensable Pentium Book*, Addison-Wesley, New York, 1995.
- Karen Miller, *An Assembly Language Introduction to Computer Architecture: Using the Intel Pentium*, Oxford University Press, New York, 1999.
- Stephen Morse, Eric Isaacson, and Douglas Albert, *The 80386/387 Architecture*, John Wiley & Sons, New York, 1987.
- NexGen Inc., *Nx586 Processor Data Book*, NexGen Inc., Milpitas, CA, 1993.
- NexGen Inc., *Nx686 Processor Data Book*, NexGen Inc., Milpitas, CA, 1994.
- Bipin Patwardhan, *Introduction to the Streaming SIMD Extensions in the Pentium III*, www.x86.org/articles/sse_pt1/simd1.htm, June, 2000.
- Peter Norton, Peter Aitken, and Richard Wilton, *PC Programmer's Bible*, Microsoft Press, Redmond, WA, 1993.
- *PharLap 386/ASM Reference Manual*, Pharlap, Cambridge MA, 1993.
- *PharLap TNT DOS-Extender Reference Manual*, Pharlap, Cambridge MA, 1995.
- Sen-Cuo Ro and Sheau-Chuen Her, *i386/i486 Advanced Programming*, Van Nostrand Reinhold, New York, 1993.
- Tom Shanley, *Protected Mode System Architecture*, Addison Wesley, NY, 1996.
- SGS-Thomson Corporation, *80486DX Processor SMM Programming Manual*, SGS-Thomson Corporation, 1995.
- Walter A. Triebel, *The 80386DX Microprocessor*, Prentice-Hall, Englewood Cliffs, NJ, 1992.
- John Wharton, *The Complete x86*, MicroDesign Resources, Sebastopol, California, 1994.

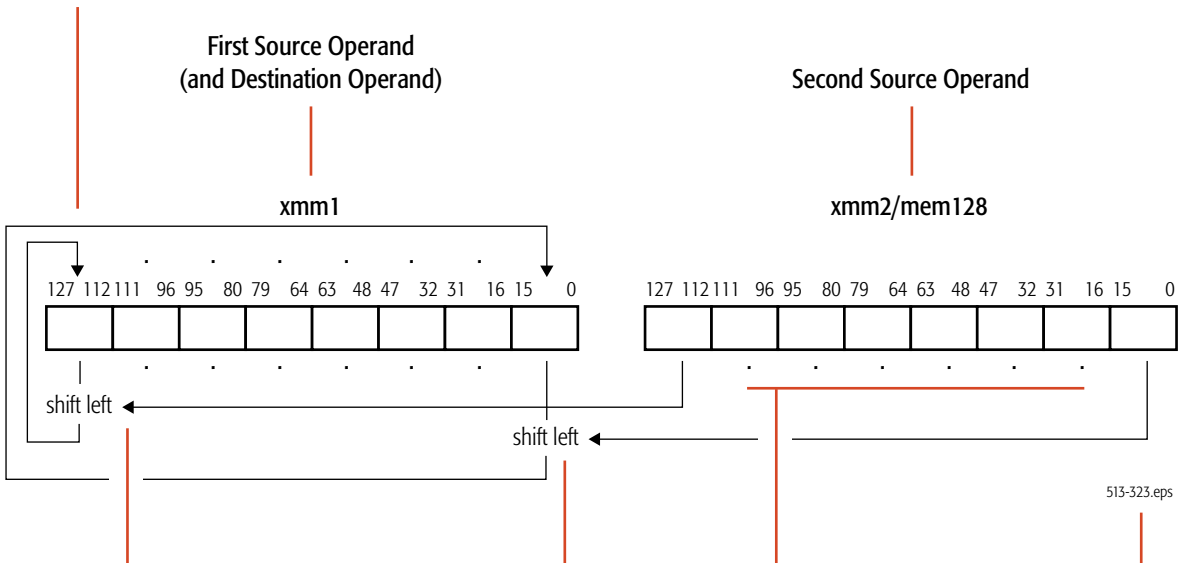
- Web sites and newsgroups:
 - www.amd.com
 - news.comp.arch
 - news.comp.lang.asm.x86
 - news.intel.microprocessors
 - news.microsoft

1 128-Bit Media Instruction Reference

This chapter describes the function, mnemonic syntax, opcodes, affected flags of the 128-bit media instructions and the possible exceptions they generate. These instructions load, store, or operate on data located in 128-bit XMM registers. Most of the instructions operate in parallel on sets of packed elements called *vectors*, although a few operate on scalars. These instructions define both integer and floating-point operations. They include the legacy SSE and SSE2 instructions.

Each instruction that performs a vector (packed) operation is illustrated with a diagram. Figure 1-1 shows the conventions used in these diagrams. The particular diagram shows the PSSLW (packed shift left logical words) instruction.

Arrowheads going to a source operand indicate the writing of the result. In this case, the result is written to the first source operand, which is also the destination operand.



Arrowheads coming from a source operand indicate that the source operand provides a *control function*. In this case, the second source operand specifies the *number* of bits to shift, and the first source operand specifies the *data* to be shifted.

Operation. In this case, a bitwise shift-left.

Ellipses indicate that the operation is repeated for each element of the source vectors. In this case, there are 8 elements in each source vector, so the operation is performed 8 times, in parallel.

File name of this figure (for documentation control)

Figure 1-1. Diagram Conventions for 128-Bit Media Instructions

Gray areas in diagrams indicate unmodified operand bits.

The 128-bit media instructions are useful in high-performance applications that operate on blocks of data. Because each instruction can independently and simultaneously perform a single operation on multiple elements of a vector, the instructions are classified as *single-instruction, multiple-data* (SIMD) instructions. A few 128-bit media instructions convert operands in XMM registers to operands in GPR, MMX™, or x87 registers (or vice versa), or save or restore XMM state.

Hardware support for a specific 128-bit media instruction depends on the presence of at least one of the following CPUID functions:

- FXSAVE and FXRSTOR, indicated by bit 24 of CPUID standard function 1 and extended function 8000_0001h.
- SSE, indicated by bit 25 of CPUID extended function 8000_0001h.
- SSE2, indicated by bit 26 of CPUID extended function 8000_0001h.

The 128-bit media instructions can be used in legacy mode or long mode. Their use in long mode is available if the following CPUID function is set:

- Long Mode, indicated by bit 29 of CPUID extended function 8000_0001h.

Compilation of 128-bit media programs for execution in 64-bit mode offers four primary advantages: access to the eight extended XMM registers (for a register set consisting of XMM0–XMM15), access to the eight extended, 64-bit general-purpose registers (for a register set consisting of GPR0–GPR15), access to the 64-bit virtual address space, and access to the RIP-relative addressing mode.

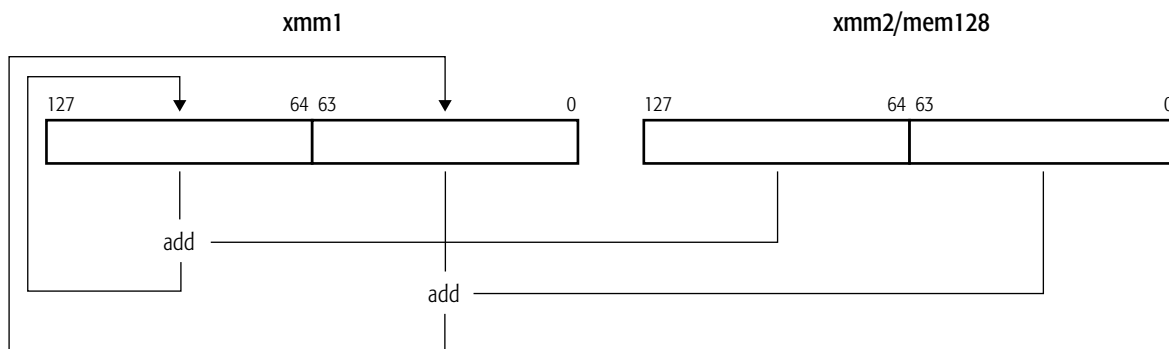
For further information, see:

- “128-Bit Media and Scientific Programming” in Volume 1.
- “Summary of Registers and Data Types” in Volume 3.
- “Notation” in Volume 3.
- “Instruction Prefixes” in Volume 3.

ADDPD**Add Packed Double-Precision Floating-Point**

The ADDPD instruction adds each packed double-precision floating-point value in the first source operand to the corresponding packed double-precision floating-point value in the second source operand and writes the result of each addition in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
ADDPD <i>xmm1, xmm2/mem128</i>	66 0F 58 /r	Adds two packed double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



addpd.eps

Related Instructions

ADDPS, ADDSD, ADDSS

rFLAGS Affected

None

MXCSR Flags Affected

None

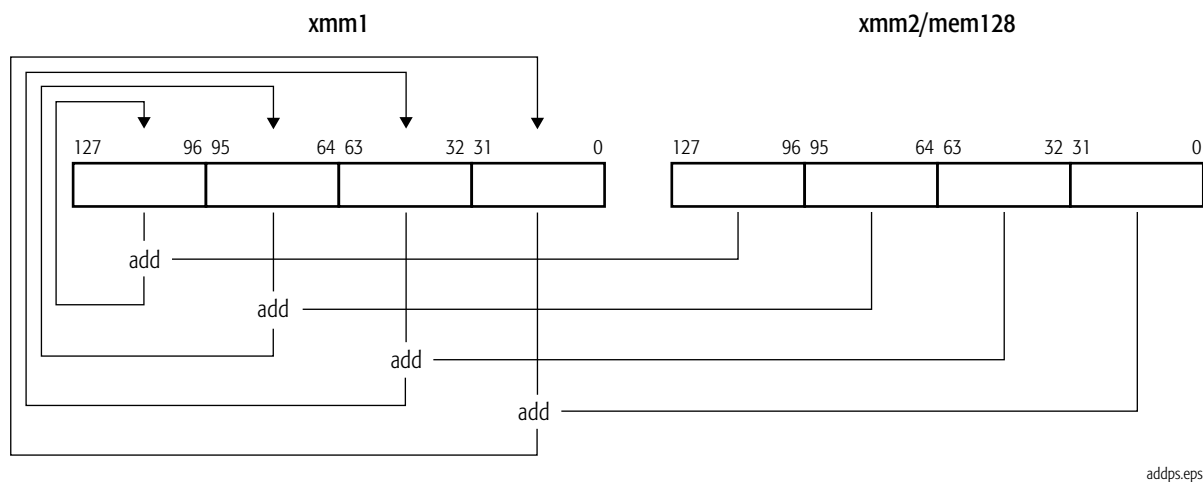
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> below for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

ADDPS**Add Packed Single-Precision Floating-Point**

The ADDPS instruction adds each packed single-precision floating-point value in the first source operand to the corresponding packed single-precision floating-point value in the second source operand and writes the result of each addition in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
ADDPS <i>xmm1, xmm2/mem128</i>	0F 58/r	Adds four packed single-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.

**Related Instructions**

ADDPD, ADDSD, ADDSS

rFLAGS Affected

None

MXCSR Flags Affected

None

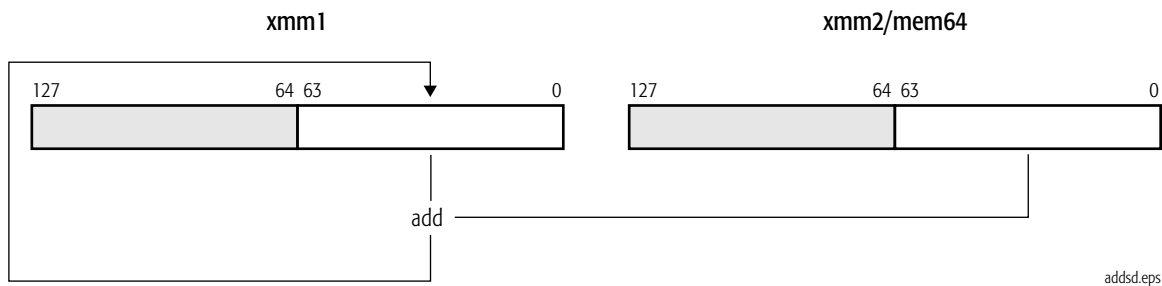
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

ADDSD**Add Scalar Double-Precision Floating-Point**

The ADDSD instruction adds the double-precision floating-point value in the low-order quadword of the first source operand to the double-precision floating-point value in the low-order quadword of the second source operand and writes the result in the low-order quadword of the destination (first source). The high-order quadword of the destination is not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 64-bit memory location.

Mnemonic	Opcode	Description
ADDSD <i>xmm1, xmm2/mem64</i>	F2 0F 58/r	Adds low-order double-precision floating-point values in an XMM register and another XMM register or 64-bit memory location and writes the result in the destination XMM register.

**Related Instructions**

ADDPD, ADDPS, ADDSS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

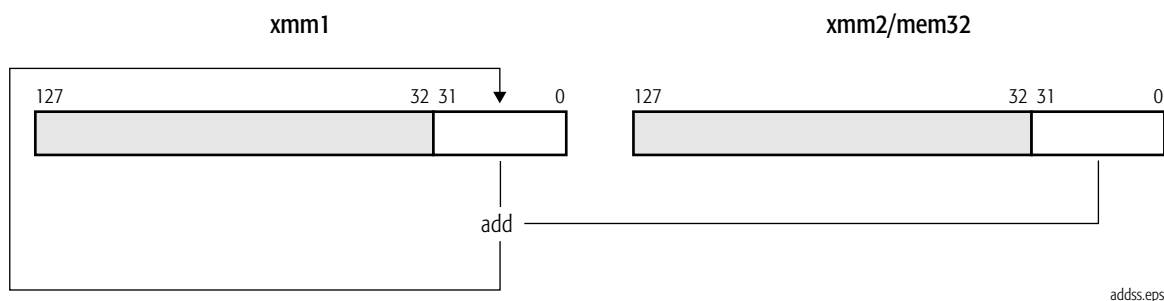
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

ADDSS**Add Scalar Single-Precision Floating-Point**

The ADDSS instruction adds the single-precision floating-point value in the low-order doubleword of the first source operand to the single-precision floating-point value in the low-order doubleword of the second source operand and writes the result in the low-order doubleword of the destination (first source). The three high-order doublewords of the destination are not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 32-bit memory location.

Mnemonic	Opcode	Description
ADDSS <i>xmm1, xmm2/mem32</i>	F3 0F 58 /r	Adds low-order single-precision floating-point values in an XMM register and another XMM register or 32-bit memory location and writes the result in the destination XMM register.

**Related Instructions**

ADDPD, ADDPS, ADDSD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

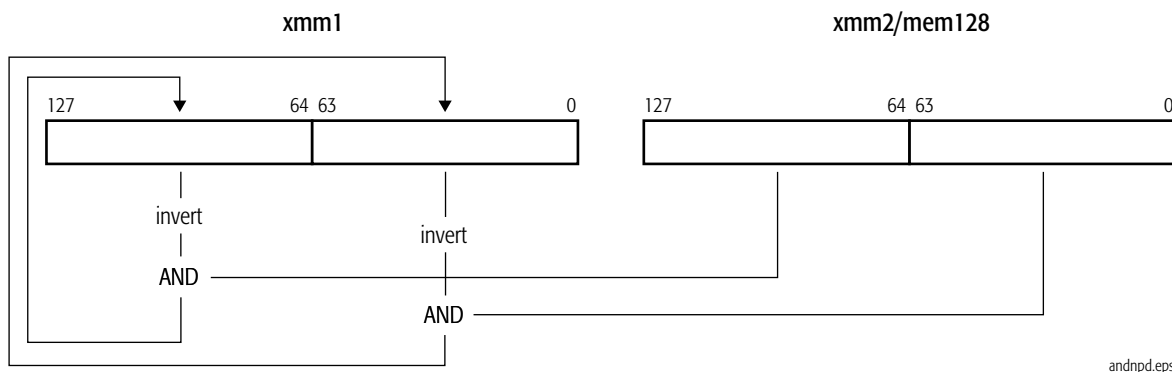
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

ANDNPD**Logical Bitwise AND NOT
Packed Double-Precision Floating-Point**

The ANDNPD instruction performs a bitwise logical AND of the two packed double-precision floating-point values in the second source operand and the one's-complement of the corresponding two packed double-precision floating-point values in the first source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
ANDNPD <i>xmm1, xmm2/mem128</i>	66 0F 55 /r	Performs bitwise logical AND NOT of two packed double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



andnpd.eps

Related Instructions

ANDNPS, ANDPD, ANDPS, ORPD, ORPS, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

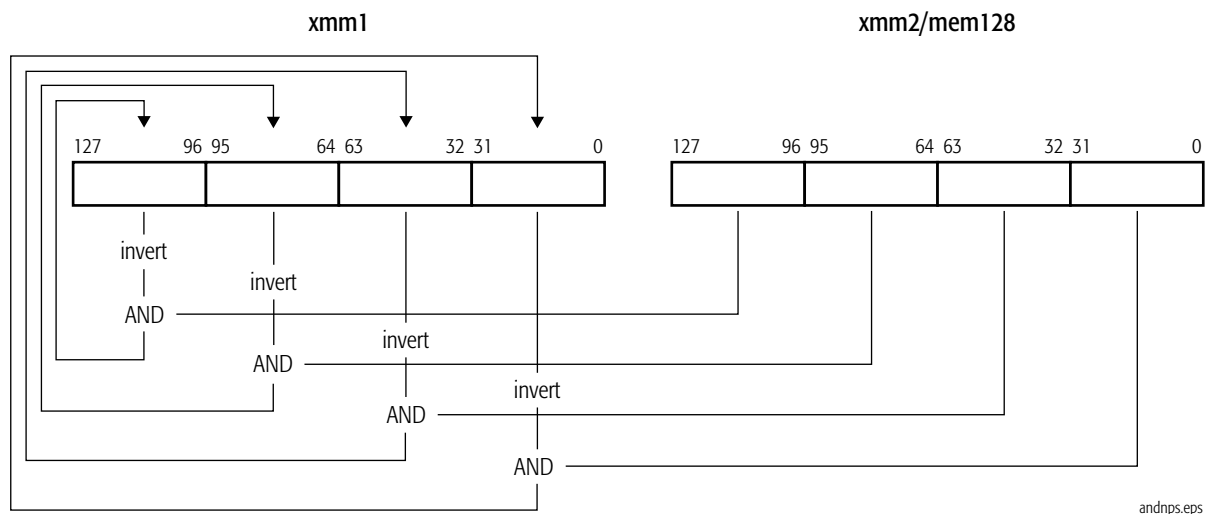
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	<p>The emulate bit (EM) of CR0 is set to 1.</p> <p>The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception.</p> <p>The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.</p> <p>The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.</p>
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		One of the data operands falls outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

ANDNPS**Logical Bitwise AND NOT
Packed Single-Precision Floating-Point**

The ANDNPS instruction performs a bitwise logical AND of the four packed single-precision floating-point values in the second source operand and the one's-complement of the corresponding four packed single-precision floating-point values in the first source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
ANDNPS <i>xmm1, xmm2/mem128</i>	0F 55/r	Performs bitwise logical AND NOT of four packed single-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



andnps.eps

Related Instructions

ANDNPD, ANDPD, ANDPS, ORPD, ORPS, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

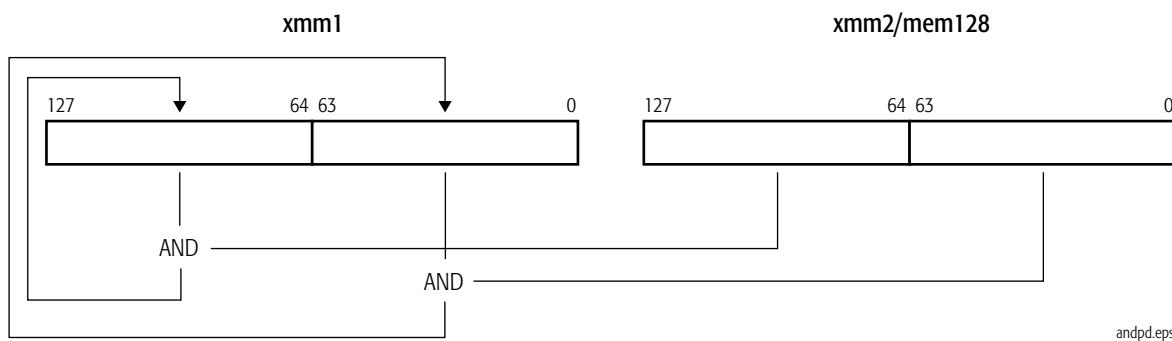
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

ANDPD**Logical Bitwise AND
Packed Double-Precision Floating-Point**

The ANDPD instruction performs a bitwise logical AND of the two packed double-precision floating-point values in the first source operand and the corresponding two packed double-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
ANDPD <i>xmm1, xmm2/mem128</i>	66 0F 54/r	Performs bitwise logical AND of two packed double-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



andpd.eps

Related Instructions

ANDNPD, ANDNPS, ANDPS, ORPD, ORPS, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

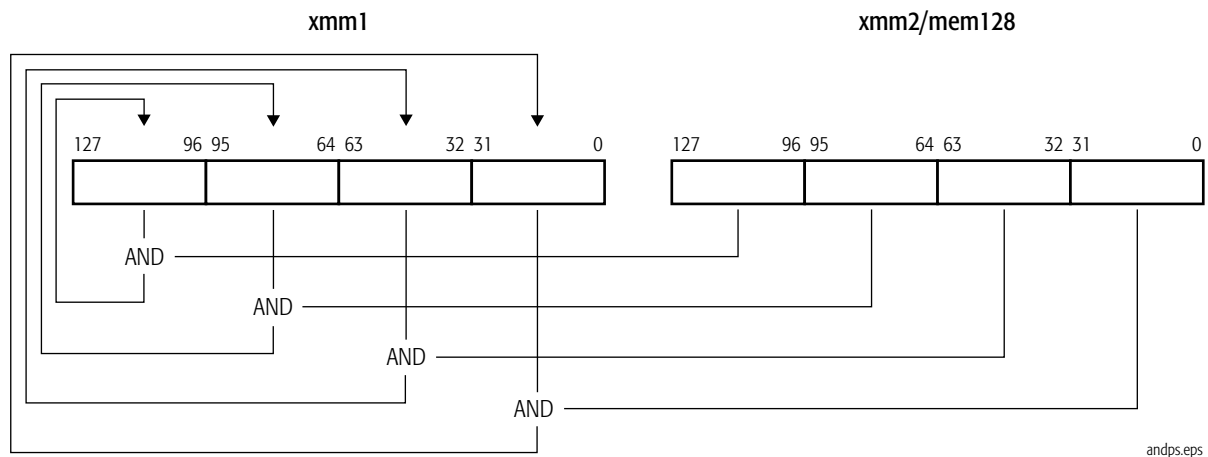
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

ANDPS**Logical Bitwise AND
Packed Single-Precision Floating-Point**

The ANDPS instruction performs a bitwise logical AND of the four packed single-precision floating-point values in the first source operand and the corresponding four packed single-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
ANDPS <i>xmm1, xmm2/mem128</i>	OF 54 /r	Performs bitwise logical AND of four packed single-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



andps.eps

Related Instructions

ANDNPD, ANDNPS, ANDPD, ORPD, ORPS, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

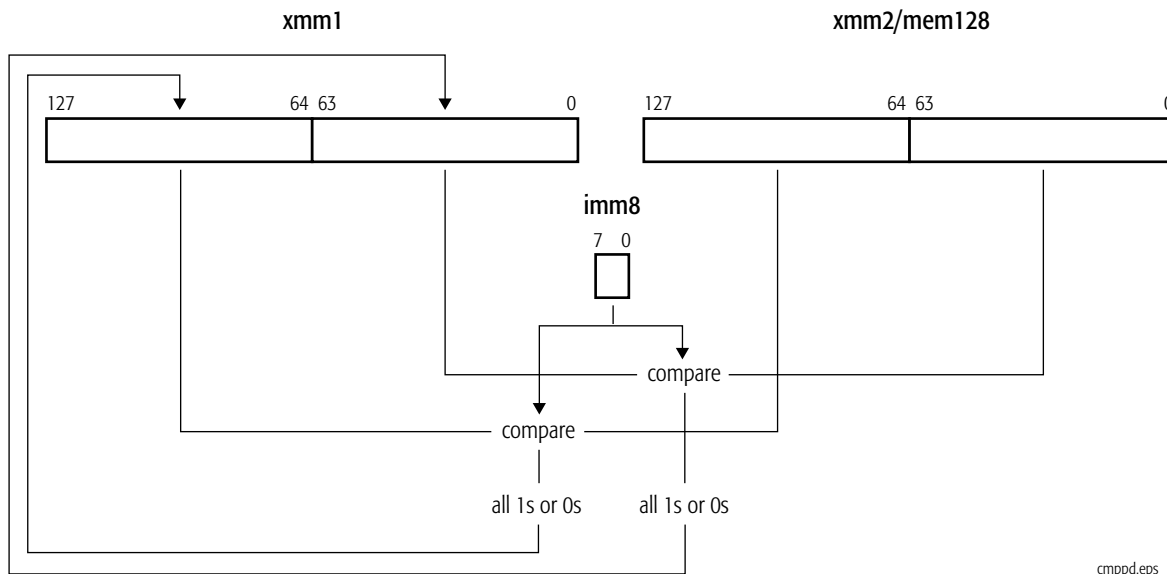
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

CMPPD**Compare Packed Double-Precision Floating-Point**

The CMPPD instruction compares each of the two packed double-precision floating-point values in the first source operand with the corresponding packed double-precision floating-point value in the second source operand and writes the result of each comparison in the corresponding 64 bits of the destination (first source). The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in Table 1-1. The result of each compare is a 64-bit value of all 1s (TRUE) or all 0s (FALSE). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
CMPPD <i>xmm1, xmm2/mem128, imm8</i>	66 0F C2 /r ib	Compares two pairs of packed double-precision floating-point values in an XMM register and an XMM register or 128-bit memory location.



Some compare operations that are not directly supported by the immediate-byte encodings can be implemented by swapping the contents of the source and destination operands and then executing the appropriate compare instruction using the swapped values. These additional compare operations are shown, together with

the directly supported compare operations, in Table 1-1. When swapping operands, the first source XMM register is overwritten by the result.

Table 1-1. Immediate Operand Values for Compare Operations

Immediate-Byte Value (bits 2–0)	Compare Operation	Result If NaN Operand	QNaN Operand Causes Invalid Operation Exception
000	Equal	FALSE	No
001	Less than	FALSE	Yes
	Greater than or equal to (uses swapped operands)	FALSE	Yes
010	Less than or equal	FALSE	Yes
	Greater than (uses swapped operands)	FALSE	Yes
011	Unordered	TRUE	No
100	Not equal	TRUE	No
101	Not less than	TRUE	Yes
	Not greater than (uses swapped operands)	TRUE	Yes
110	Not less than or equal	TRUE	Yes
	Not greater than or equal (uses swapped operands)	TRUE	Yes
111	Ordered	FALSE	No

Related Instructions

CMPPS, CMPSD, CMPSS, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Note: A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.																	

Exceptions

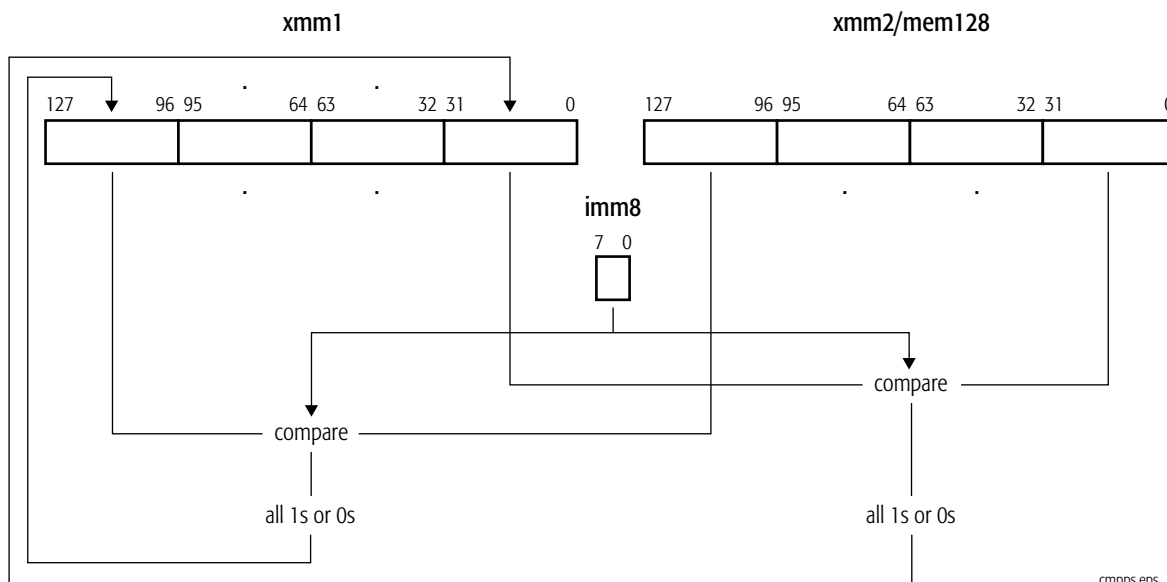
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value. For some compares, the source operand is a QNaN value (refer to Table 1-1 on page 23).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

CMPPS**Compare Packed Single-Precision Floating-Point**

The CMPPS instruction compares each of the four packed single-precision floating-point values in the first source operand with the corresponding packed single-precision floating-point value in the second source operand and writes the result of each comparison in the corresponding 32 bits of the destination (first source). The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in Table 1-1 on page 23. The result of each compare is a 32-bit value of all 1s (TRUE) or all 0s (FALSE). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
CMPPS <i>xmm1, xmm2/mem128, imm8</i>	0F C2/r ib	Compares four pairs of packed single-precision floating-point values in an XMM register and an XMM register or 64-bit memory location.



cmpps.eps

Some compare operations that are not directly supported by the immediate-byte encodings can be implemented by swapping the contents of the source and destination operands and then executing the appropriate compare instruction using

the swapped values. These additional compare operations are shown in Table 1-1 on page 23. When swapping operands, the first source XMM register is overwritten by the result.

Related Instructions

CMPPD, CMPSD, CMPSS, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ		RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																	M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

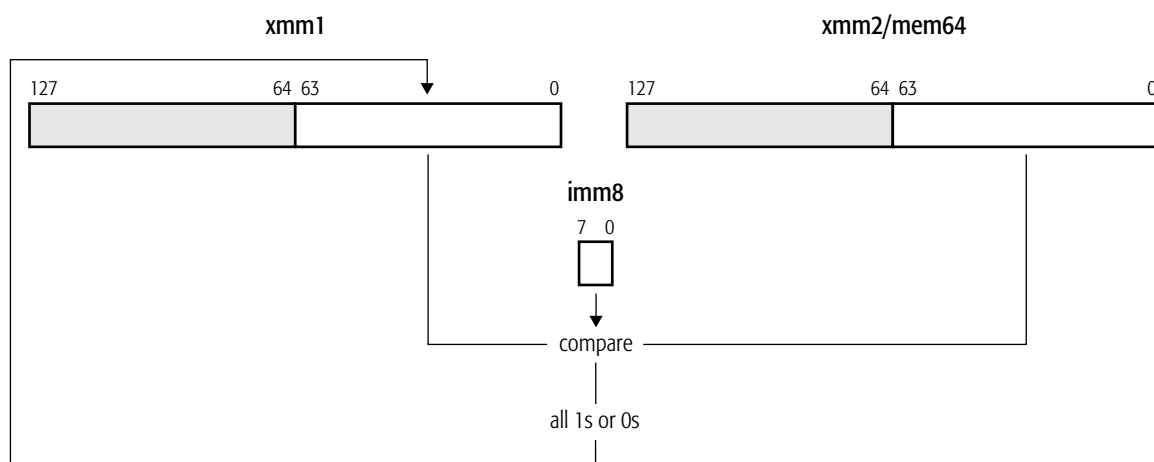
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value. For some compares, the source operand is a QNaN value (refer to Table 1-1 on page 23).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

CMPSD**Compare Scalar Double-Precision Floating-Point**

The CMPSD instruction compares the double-precision floating-point value in the low-order 64 bits of the first source operand with the double-precision floating-point value in the low-order 64 bits of the second source operand and writes the result in the low-order 64 bits of the destination (first source). The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in Table 1-1 on page 23. The result of the compare is a 64-bit value of all 1s (TRUE) or all 0s (FALSE). The first source/destination operand is an XMM register. The second source operand is another XMM register or 64-bit memory location. The high-order 64 bits of the destination XMM register are not modified.

Mnemonic	Opcode	Description
CMPSD <i>xmm1, xmm2/mem64, imm8</i>	F2 0F C2 /r ib	Compares double-precision floating-point values in an XMM register and an XMM register or 64-bit memory location.



cmpsd.eps

Some compare operations that are not directly supported by the immediate-byte encodings can be implemented by swapping the contents of the source and destination operands and then executing the appropriate compare instruction using the swapped values. These additional compare operations are shown in Table 1-1 on page 23. When swapping operands, the first source XMM register is overwritten by the result.

This CMPSD instruction should not be confused with the same-mnemonic CMPSD (compare strings by doubleword) instruction in the general-purpose instruction set. Assemblers can distinguish the instructions by the number and type of operands.

Related Instructions

CMPPD, CMPPS, CMPSS, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

	FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE	
															M	M	
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Note: A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.																	

Exceptions

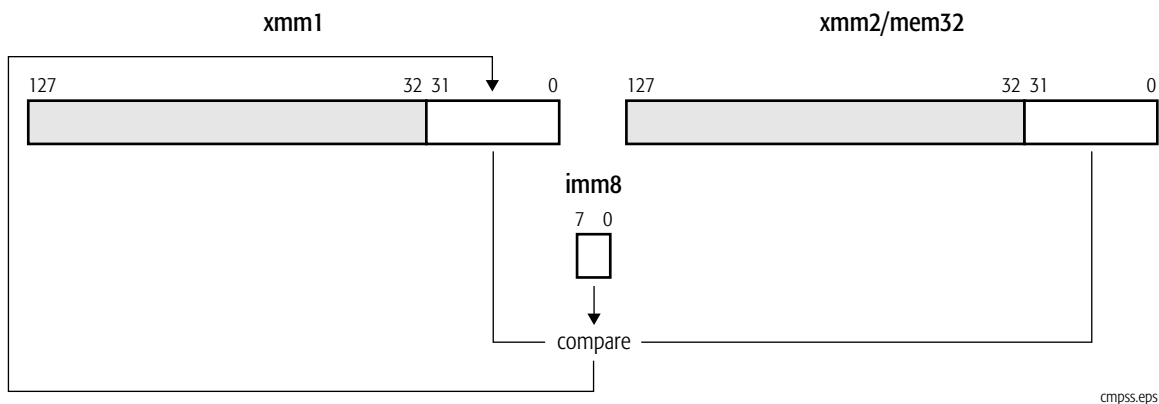
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value. For some compares, the source operand is a QNaN value (refer to Table 1-1 on page 23).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

CMPSS**Compare Scalar Single-Precision Floating-Point**

The CMPSS instruction compares the single-precision floating-point value in the low-order 32 bits of the first source operand with the single-precision floating-point value in the low-order 32 bits of the second source operand and writes the result in the low-order 32 bits of the destination (first source). The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in Table 1-1 on page 23. The result of the compare is a 32-bit value of all 1s (TRUE) or all 0s (FALSE). The first source/destination operand is an XMM register. The second source operand is another XMM register or 32-bit memory location. The three high-order doublewords of the destination XMM register are not modified.

Mnemonic	Opcode	Description
CMPSS <i>xmm1, xmm2/mem32, imm8</i>	F3 0F C2 /r ib	Compares single-precision floating-point values in an XMM register and an XMM register or 32-bit memory location.



Some compare operations that are not directly supported by the immediate-byte encodings can be implemented by swapping the contents of the source and destination operands and then executing the appropriate compare instruction using the swapped values. These additional compare operations are shown in Table 1-1 on page 23. When swapping operands, the first source XMM register is overwritten by the result.

Related Instructions

CMPPD, CMPPS, CMPSD, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:

A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

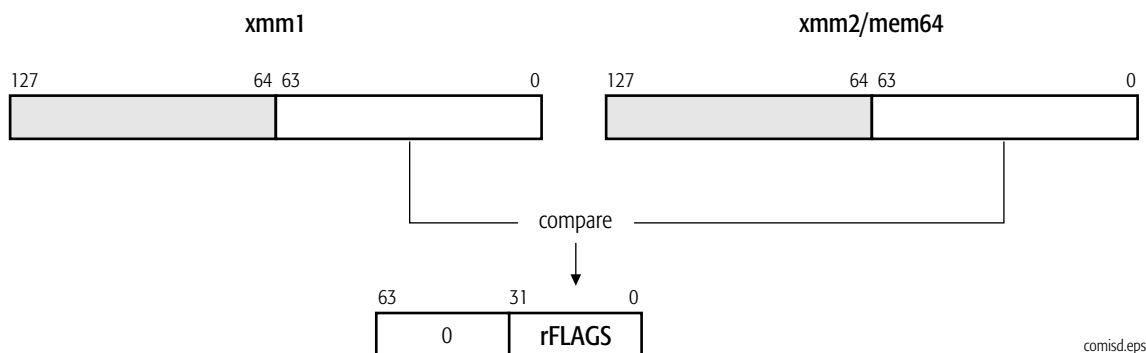
Exception	Real	Virtual 8086	Protected	Cause of Exception
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value. For some compares, the source operand is a QNaN value (refer to Table 1-1 on page 23).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

COMISD Compare Ordered Scalar Double-Precision Floating-Point

The COMISD instruction performs an ordered compare of the double-precision floating-point value in the low-order 64 bits of an XMM register with the double-precision floating-point value in the low-order 64 bits of another XMM register or a 64-bit memory location and sets the ZF, PF, and CF bits in the rFLAGS register to reflect the result of the compare. The result is unordered if one or both of the operand values is a NaN. The OF, AF, and SF bits in rFLAGS are set to zero.

If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

Mnemonic	Opcode	Description
COMISD <i>xmm1, xmm2/mem64</i>	66 0F 2F /r	Compares double-precision floating-point values in an XMM register and an XMM register or 64-bit memory location. Sets rFLAGS.



Result of Compare	ZF	PF	CF
Unordered	1	1	1
Greater Than	0	0	0
Less Than	0	0	1
Equal	1	0	0

Related Instructions

CMPPD, CMPPS, CMPSD, CMPSS, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

ID	VIP	VIF	AC	VM	RF	NT	IOPL	OF	DF	IF	TF	SF	ZF	AF	PF	CF
								0				0	M	0	M	M
21	20	19	18	17	16	14	13–12	11	10	9	8	7	6	4	2	0

Note:
 Bits 31–22, 15, 5, 3, and 1 are reserved. A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank.
 If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

MXCSR Flags Affected

		FZ		RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																	M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Note: A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.																		

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).

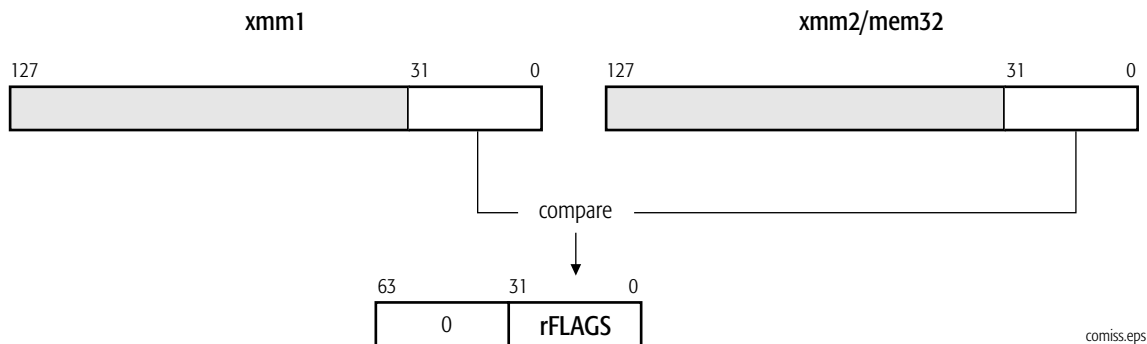
Exception	Real	Virtual 8086	Protected	Cause of Exception
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand is an SNaN or QNaN value or an unsupported format.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

COMISS**Compare Ordered Scalar Single-Precision Floating-Point**

The COMISS instruction performs an ordered compare of the single-precision floating-point value in the low-order 32 bits of an XMM register with the single-precision floating-point value in the low-order 32 bits of another XMM register or a 32-bit memory location and sets the ZF, PF, and CF bits in the rFLAGS register to reflect the result of the compare. The result is unordered if one or both of the operand values is a NaN. The OF, AF, and SF bits in rFLAGS are set to zero.

If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

Mnemonic	Opcode	Description
COMISS <i>xmm1, xmm2/mem32</i>	0F 2F /r	Compares single-precision floating-point values in an XMM register and an XMM register or 32-bit memory location. Sets rFLAGS.



comiss.eps

Result of Compare	ZF	PF	CF
Unordered	1	1	1
Greater Than	0	0	0
Less Than	0	0	1
Equal	1	0	0

Related Instructions

CMPPD, CMPPS, CMPSD, CMPSS, COMISD, UCOMISD, UCOMISS

rFLAGS Affected

ID	VIP	VIF	AC	VM	RF	NT	IOPL	OF	DF	IF	TF	SF	ZF	AF	PF	CF
								0				0	M	0	M	M
21	20	19	18	17	16	14	13–12	11	10	9	8	7	6	4	2	0

Note:
 Bits 31–22, 15, 5, 3, and 1 are reserved. A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank.
 If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Note: A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.																	

Exceptions

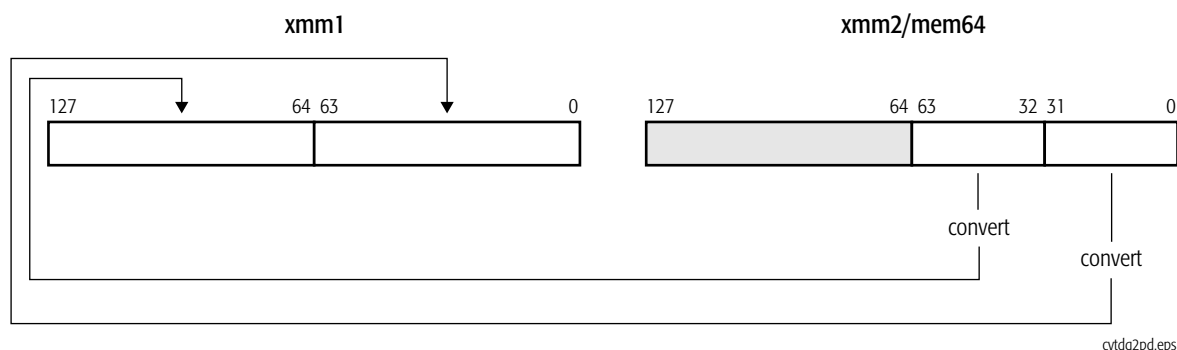
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).

Exception	Real	Virtual 8086	Protected	Cause of Exception
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand is an SNaN or QNaN value or an unsupported format.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

CVTDQ2PD**Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point**

The CVTDQ2PD instruction converts two packed 32-bit signed integer values in the low-order 64 bits of an XMM register or a 64-bit memory location to two packed double-precision floating-point values and writes the converted values in another XMM register.

Mnemonic	Opcode	Description
CVTDQ2PD <i>xmm1, xmm2/mem64</i>	F3 0F E6	Converts packed doubleword signed integers in an XMM register or 64-bit memory location to double-precision floating-point values in the destination XMM register.

**Related Instructions**

CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

None

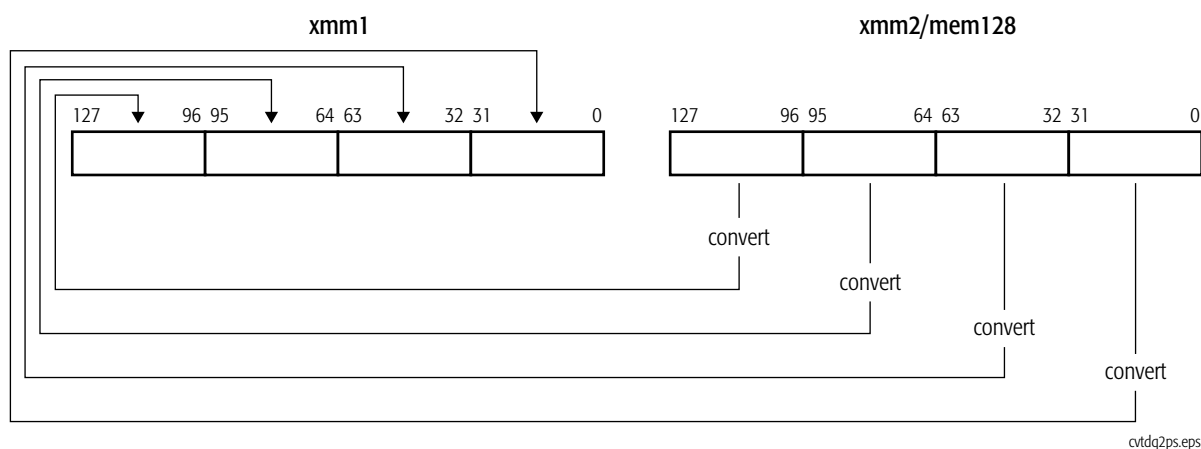
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

CVTDQ2PS**Convert Packed Doubleword Integers to Packed Single-Precision Floating-Point**

The CVTDQ2PS instruction converts four packed 32-bit signed integer values in an XMM register or a 128-bit memory location to four packed single-precision floating-point values and writes the converted values in another XMM register. If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register.

Mnemonic	Opcode	Description
CVTDQ2PS <i>xmm1, xmm2/mem128</i>	0F 5B /r	Converts packed doubleword integer values in an XMM register or 128-bit memory location to packed single-precision floating-point values in the destination XMM register.

**Related Instructions**

CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M					
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

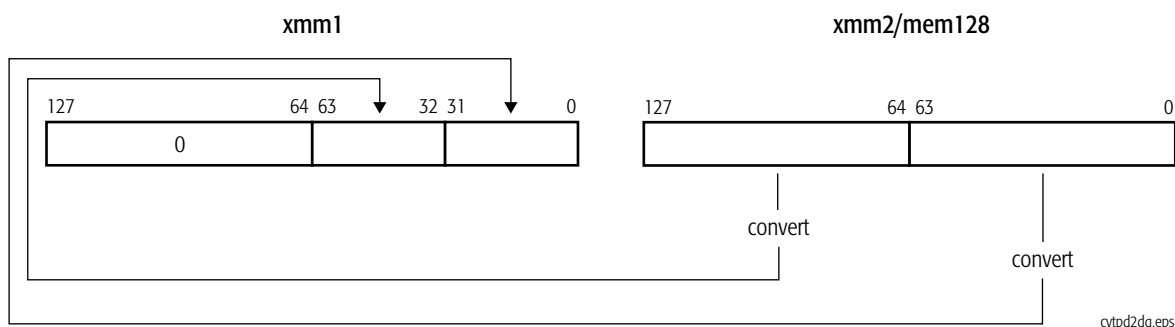
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTPD2DQ**Convert Packed Double-Precision Floating-Point to Packed Doubleword Integers**

The CVTPD2DQ instruction converts two packed double-precision floating-point values in an XMM register or a 128-bit memory location to two packed 32-bit signed integers and writes the converted values in the low-order 64 bits of another XMM register. The high-order 64 bits in the destination XMM register are cleared to all 0s.

Mnemonic	Opcode	Description
CVTPD2DQ <i>xmm1, xmm2/mem128</i>	F2 0F E6	Converts packed double-precision floating-point values in an XMM register or 128-bit memory location to packed doubleword integers in the destination XMM register.



If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

Related Instructions

CVTDQ2PD, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

	FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

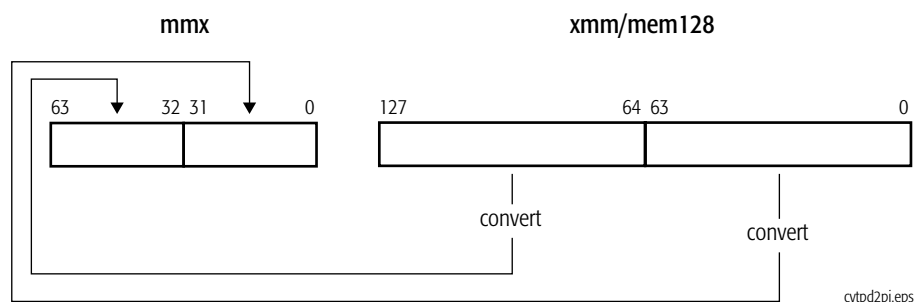
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTPD2PI**Convert Packed Double-Precision Floating-Point to Packed Doubleword Integers**

The CVTPD2PI instruction converts two packed double-precision floating-point values in an XMM register or a 128-bit memory location to two packed 32-bit signed integer values and writes the converted values in an MMX register.

Mnemonic	Opcode	Description
CVTPD2PI <i>mmx, xmm2/mem128</i>	66 0F 2D /r	Converts packed double-precision floating-point values in an XMM register or 128-bit memory location to packed doubleword integers values in the destination MMX™ register.



If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

Execution of this instruction causes all fields in the x87 tag word to be set according to their corresponding data, the top-of-stack-pointer bit (TOP) in the x87 status word to be cleared to 0, and any pending x87 exceptions are handled before this instruction is executed. For details, see “Actions Taken on Executing 64-Bit Media Instructions” in Volume 1.

Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

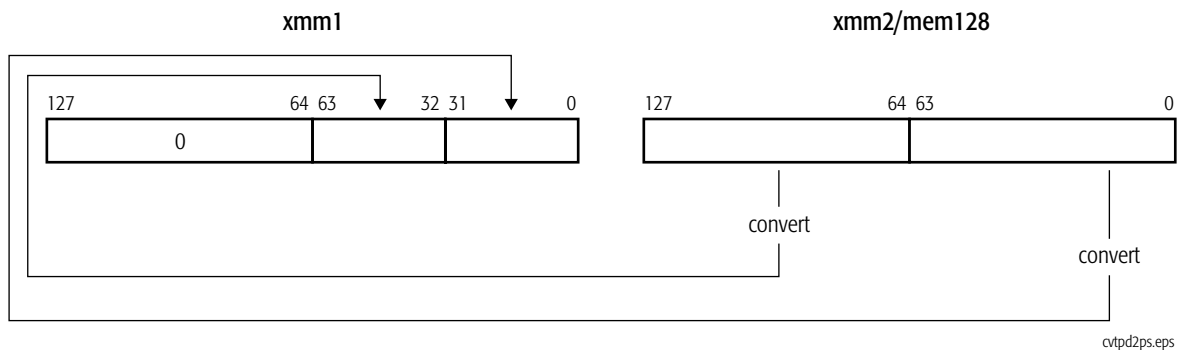
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception is pending due to an x87 floating-point instruction.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTPD2PS**Convert Packed Double-Precision Floating-Point to Packed Single-Precision Floating-Point**

The CVTPD2PS instruction converts two packed double-precision floating-point values in an XMM register or a 128-bit memory location to two packed single-precision floating-point values and writes the converted values in the low-order 64 bits of another XMM register. The high-order 64 bits in the destination XMM register are cleared to all 0s.

Mnemonic	Opcode	Description
CVTPD2PS <i>xmm1, xmm2/mem128</i>	66 0F 5A/ <i>r</i>	Converts packed double-precision floating-point values in an XMM register or 128-bit memory location to packed single-precision floating-point values in the destination XMM register.



If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register.

Related Instructions

CVTPS2PD, CVTSD2SS, CVTSS2SD

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M		M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

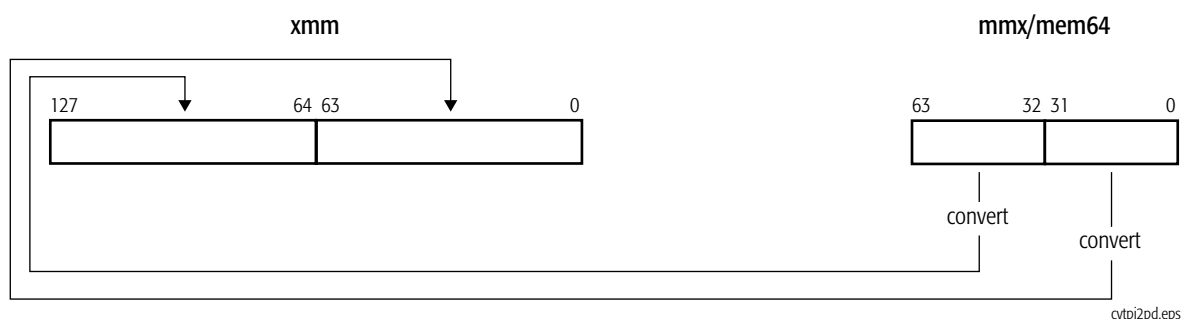
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTPI2PD**Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point**

The CVTPI2PD instruction converts two packed 32-bit signed integer values in an MMX register or a 64-bit memory location to two double-precision floating-point values and writes the converted values in an XMM register.

Mnemonic	Opcode	Description
CVTPI2PD <i>xmm, mmx/mem64</i>	66 0F 2A/r	Converts two packed doubleword integer values in an MMX™ register or 64-bit memory location to two packed double-precision floating-point values in the destination XMM register.



Execution of this instruction causes all fields in the x87 tag word to be set according to their corresponding data, the top-of-stack-pointer bit (TOP) in the x87 status word to be cleared to 0, and any pending x87 exceptions are handled before this instruction is executed. For details, see “Actions Taken on Executing 64-Bit Media Instructions” in Volume 1.

Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

None

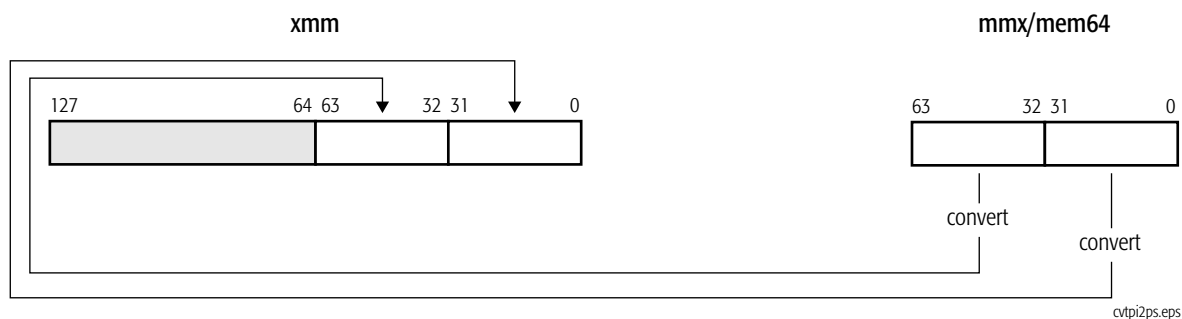
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception is pending due to an x87 floating-point instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTPI2PS**Convert Packed Doubleword Integers to Packed Single-Precision Floating-Point**

The CVTPI2PS instruction converts two packed 32-bit signed integer values in an MMX register or a 64-bit memory location to two single-precision floating-point values and writes the converted values in the low-order 64 bits of an XMM register. The high-order 64 bits of the XMM register are not modified.

Mnemonic	Opcode	Description
CVTPI2PS <i>xmm, mmx/mem64</i>	0F 2A/ <i>r</i>	Converts packed doubleword integer values in an MMX™ register or 64-bit memory location to single-precision floating-point values in the destination XMM register.



Execution of this instruction causes all fields in the x87 tag word to be set according to their corresponding data, the top-of-stack-pointer bit (TOP) in the x87 status word to be cleared to 0, and any pending x87 exceptions are handled before this instruction is executed. For details, see “Actions Taken on Executing 64-Bit Media Instructions” in Volume 1.

Related Instructions

CVTDQ2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

	FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

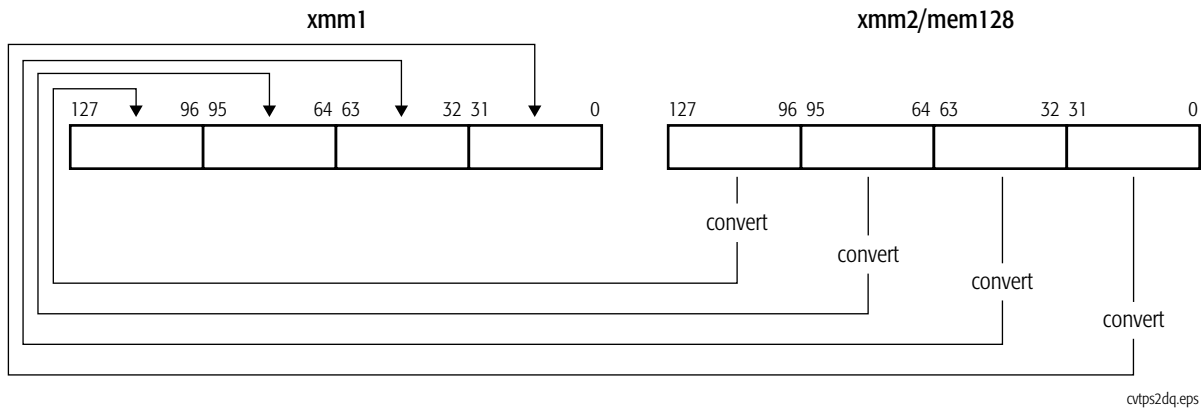
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception is pending due to an x87 floating-point instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTPS2DQ**Convert Packed Single-Precision Floating-Point to Packed Doubleword Integers**

The CVTPS2DQ instruction converts four packed single-precision floating-point values in an XMM register or a 128-bit memory location to four packed 32-bit signed integer values and writes the converted values in another XMM register.

Mnemonic	Opcode	Description
CVTPS2DQ <i>xmm1, xmm2/mem128</i>	66 0F 5B /r	Converts four packed single-precision floating-point values in an XMM register or 128-bit memory location to four packed doubleword integers in the destination XMM register.



If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

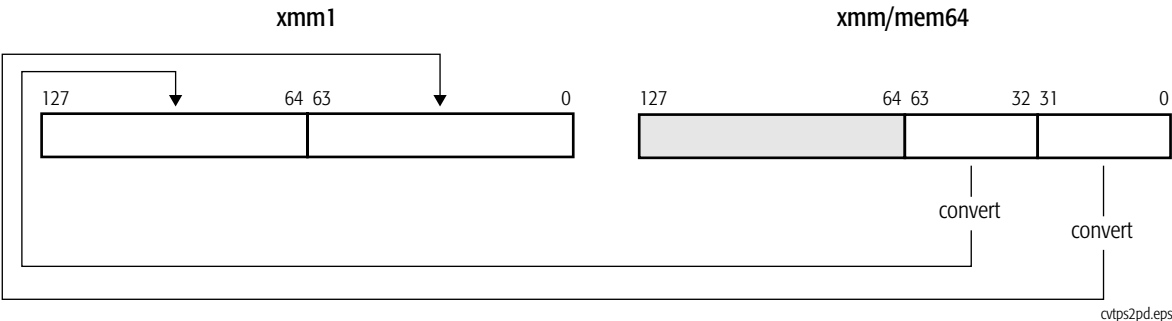
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVT_{PS}2PD

Convert Packed Single-Precision Floating-Point
to Packed Double-Precision Floating-Point

The CVT_{PS}2PD instruction converts two packed single-precision floating-point values in the low-order 64 bits of an XMM register or a 64-bit memory location to two packed double-precision floating-point values and writes the converted values in another XMM register.

Mnemonic	Opcode	Description
CVT _{PS} 2PD <i>xmm1, xmm2/mem64</i>	0F 5A/ <i>r</i>	Converts packed single-precision floating-point values in an XMM register or 64-bit memory location to packed double-precision floating-point values in the destination XMM register.



Related Instructions

CVTPD2PS, CVTSD2SS, CVTSS2SD

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Note: A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.																	

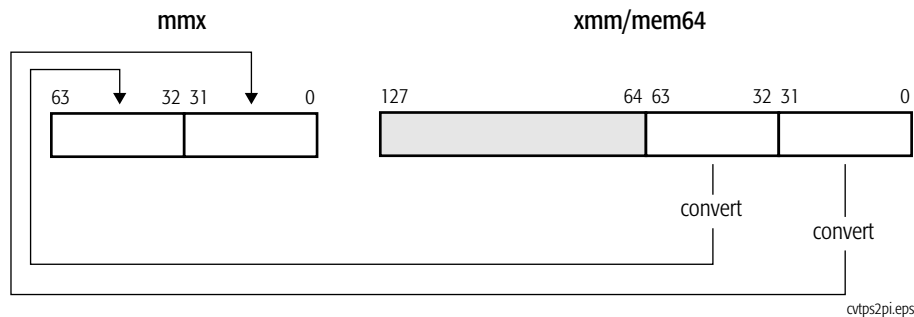
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

CVTSP2PI**Convert Packed Single-Precision Floating-Point to Packed Doubleword Integers**

The CVTSP2PI instruction converts two packed single-precision floating-point values in the low-order 64 bits of an XMM register or a 64-bit memory location to two packed 32-bit signed integers and writes the converted values in an MMX register.

Mnemonic	Opcode	Description
CVTSP2PI <i>mmx, xmm/mem64</i>	0F 2D /r	Converts packed single-precision floating-point values in an XMM register or 64-bit memory location to packed doubleword integers in the destination MMX™ register.



If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

Execution of this instruction causes all fields in the x87 tag word and the top-of-stack-pointer bit (TOP) in the x87 status word to be cleared to 0, and any pending x87 exceptions are handled before this instruction is executed. For details, see “Actions Taken on Executing 64-Bit Media Instructions” in Volume 1.

Related Instructions

CVTDQ2PS, CVTPI2PS, CVTSP2DQ, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

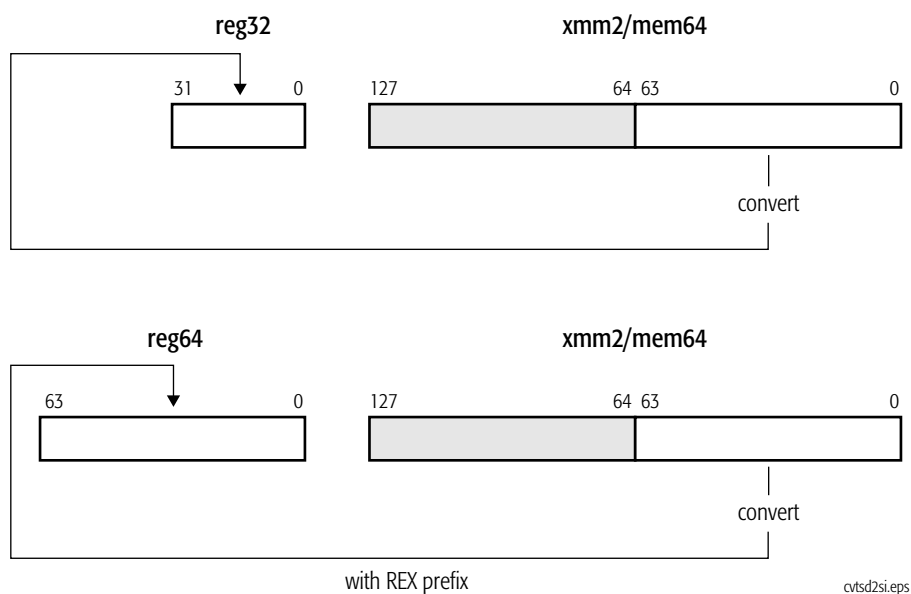
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception is pending due to an x87 floating-point instruction.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTSD2SI**Convert Scalar Double-Precision Floating-Point to Signed Doubleword or Quadword Integer**

The CVTSD2SI instruction converts a scalar double-precision floating-point value in the low-order 64 bits of an XMM register or a 64-bit memory location to a 32-bit or 64-bit signed integer and writes the converted value in a general-purpose register.

Mnemonic	Opcode	Description
CVTSD2SI <i>reg32, xmm/mem64</i>	F2 0F 2D /r	Converts a packed double-precision floating-point value in an XMM register or 64-bit memory location to a doubleword integer in a general-purpose register.
CVTSD2SI <i>reg64, xmm/mem64</i>	F2 0F 2D /r	Converts a packed double-precision floating-point value in an XMM register or 64-bit memory location to a quadword integer in a general-purpose register.



If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$) or quadword value (-2^{63} to $+2^{63} - 1$), the instruction returns the indefinite integer value (8000_0000h for 32-bit integers,

8000_0000_0000_0000h for 64-bit integers) when the invalid-operation exception (IE) is masked.

Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Note: A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.																	

Exceptions

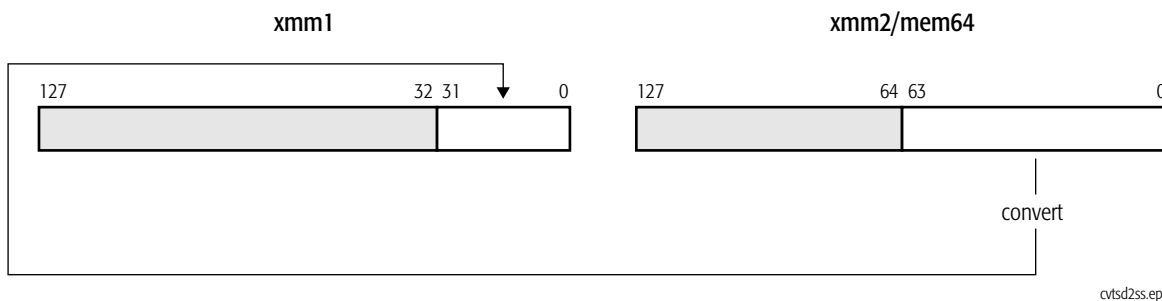
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTSD2SS**Convert Scalar Double-Precision Floating-Point to Scalar Single-Precision Floating-Point**

The CVTSD2SS instruction converts a scalar double-precision floating-point value in the low-order 64 bits of an XMM register or a 64-bit memory location to a single-precision floating-point value and writes the converted value in the low-order 64 bits of another XMM register. The three high-order doublewords in the destination XMM register are not modified. If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register.

Mnemonic	Opcode	Description
CVTSD2SS <i>xmm1, xmm2/mem64</i>	F2 0F 5A/r	Converts a scalar double-precision floating-point value in an XMM register or 64-bit memory location to a scalar single-precision floating-point value in the destination XMM register.

**Related Instructions**

CVTPD2PS, CVTPS2PD, CVTSS2SD

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M		M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

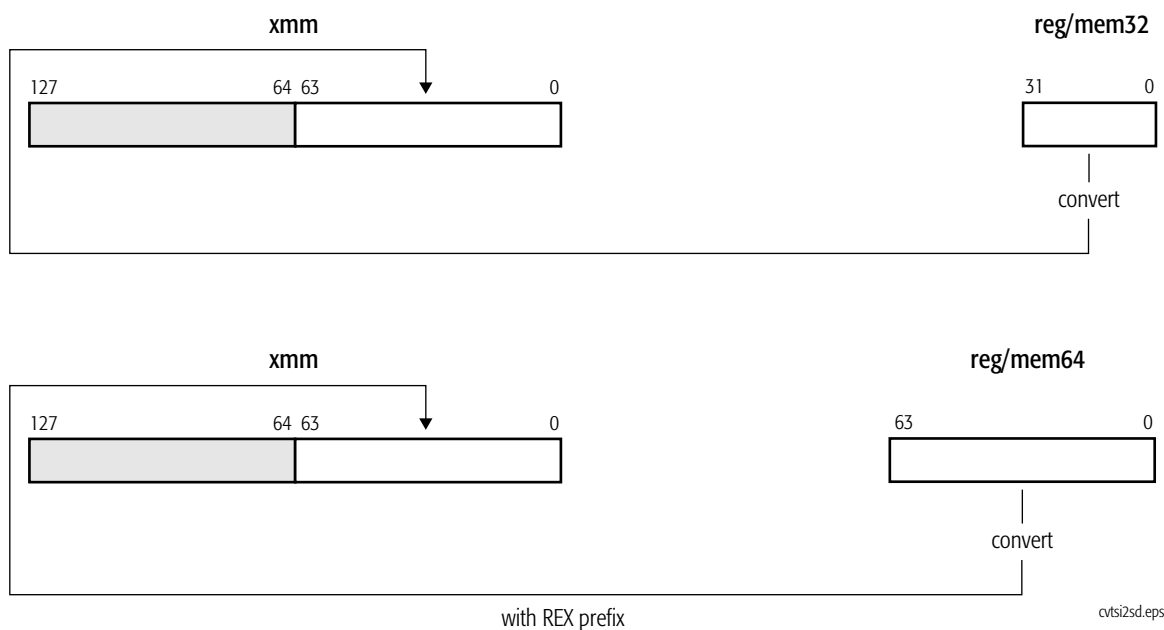
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTSI2SD**Convert Signed Doubleword or Quadword Integer to Scalar Double-Precision Floating-Point**

The CVTSI2SD instruction converts a 32-bit or 64-bit signed integer value in a general-purpose register or memory location to a double-precision floating-point value and writes the converted value in the low-order 64 bits of an XMM register. The high-order 64 bits in the destination XMM register are not modified.

Mnemonic	Opcode	Description
CVTSI2SD <i>xmm, reg/mem32</i>	F2 0F 2A/r	Converts a doubleword integer in a general-purpose register or 32-bit memory location to a double-precision floating-point value in the destination XMM register.
CVTSI2SD <i>xmm, reg/mem64</i>	F2 0F 2A/r	Converts a quadword integer in a general-purpose register or 64-bit memory location to a double-precision floating-point value in the destination XMM register.



If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register.

Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

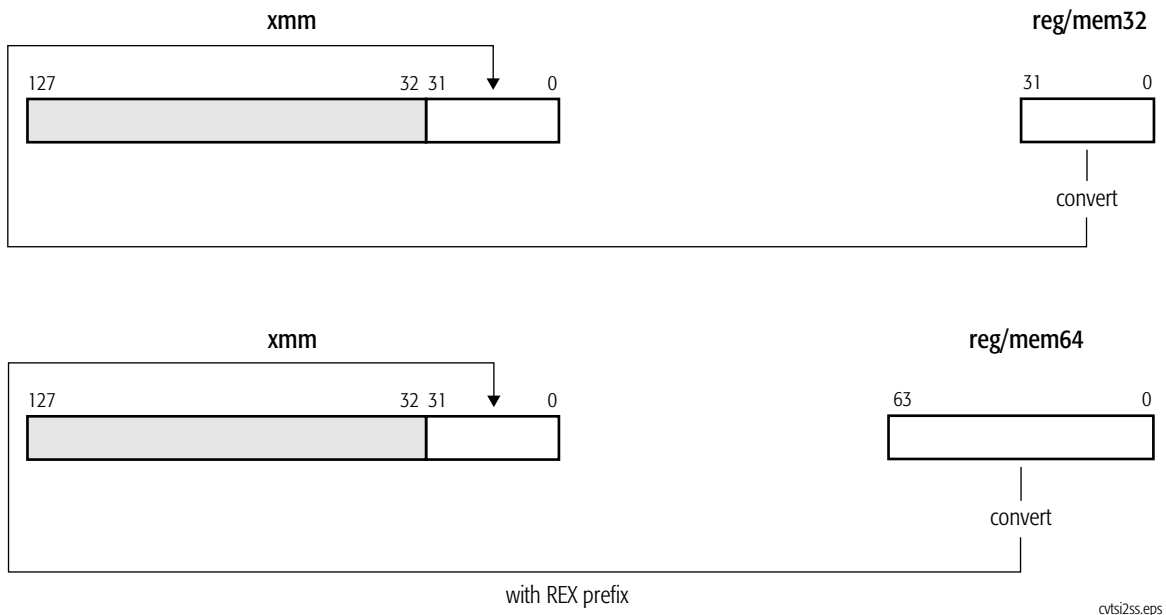
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTSI2SS**Convert Signed Doubleword or Quadword Integer to Scalar Single-Precision Floating-Point**

The CVTSI2SS instruction converts a 32-bit or 64-bit signed integer value in a general-purpose register or memory location to a single-precision floating-point value and writes the converted value in the low-order 32 bits of an XMM register. The three high-order doublewords in the destination XMM register are not modified.

Mnemonic	Opcode	Description
CVTSI2SS <i>xmm, reg/mem32</i>	F3 0F 2A/r	Converts a doubleword integer in a general-purpose register or 32-bit memory location to a single-precision floating-point value in the destination XMM register.
CVTSI2SS <i>xmm, reg/mem64</i>	F3 0F 2A/r	Converts a quadword integer in a general-purpose register or 64-bit memory location to a single-precision floating-point value in the destination XMM register.



If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register.

Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

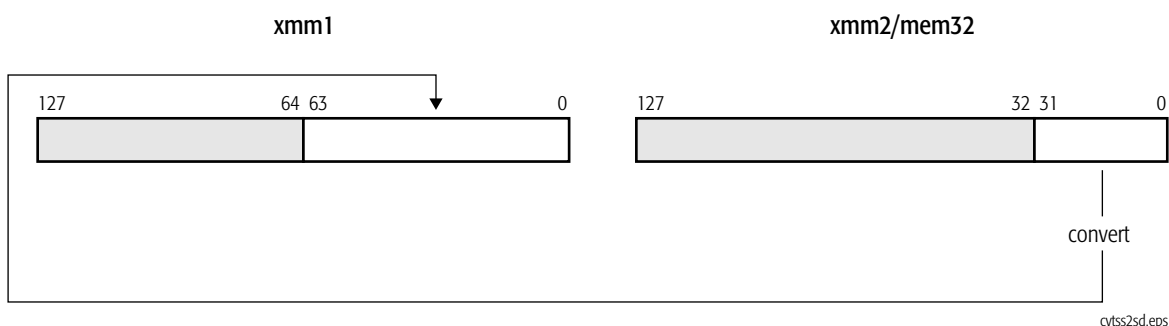
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTSS2SD**Convert Scalar Single-Precision Floating-Point to Scalar Double-Precision Floating-Point**

The CVTSS2SD instruction converts a single-precision floating-point value in the low-order 32 bits of an XMM register or a 32-bit memory location to a double-precision floating-point value and writes the converted value in the low-order 64 bits of another XMM register. The high-order 64 bits in the destination XMM register are not modified.

Mnemonic	Opcode	Description
CVTSS2SD <i>xmm1, xmm2/mem32</i>	F3 0F 5A/ <i>r</i>	Converts scalar single-precision floating-point value in an XMM register or 32-bit memory location to double-precision floating-point value in the destination XMM register.

**Related Instructions**

CVTPD2PS, CVTPS2PD, CVTSD2SS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

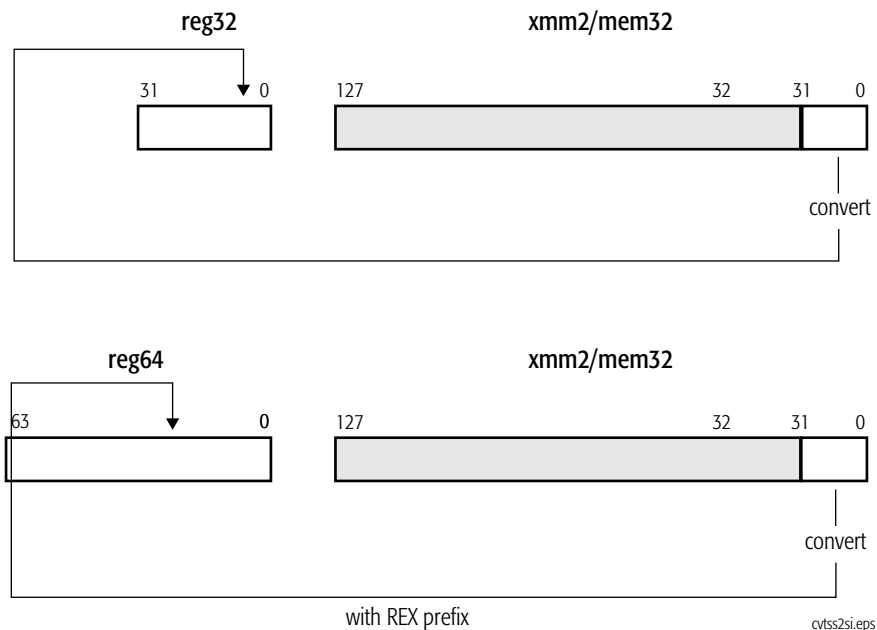
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

CVTSS2SI**Convert Scalar Single-Precision Floating-Point to Signed Doubleword or Quadword Integer**

The CVTSS2SI instruction converts a single-precision floating-point value in the low-order 32 bits of an XMM register or a 32-bit memory location to a 32-bit or 64-bit signed integer value and writes the converted value in a general-purpose register.

Mnemonic	Opcode	Description
CVTSS2SI <i>reg32, xmm2/mem32</i>	F3 0F 2D /r	Converts a single-precision floating-point value in an XMM register or 32-bit memory location to a doubleword integer value in a general-purpose register.
CVTSS2SI <i>reg64, xmm2/mem32</i>	F3 0F 2D /r	Converts a single-precision floating-point value in an XMM register or 32-bit memory location to a quadword integer value in a general-purpose register.



If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$) or quadword value (-2^{63} to $+2^{63} - 1$), the instruction returns the indefinite integer value (8000_0000h for 32-bit integers,

8000_0000_0000_0000h for 64-bit integers) when the invalid-operation exception (IE) is masked.

Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

	FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

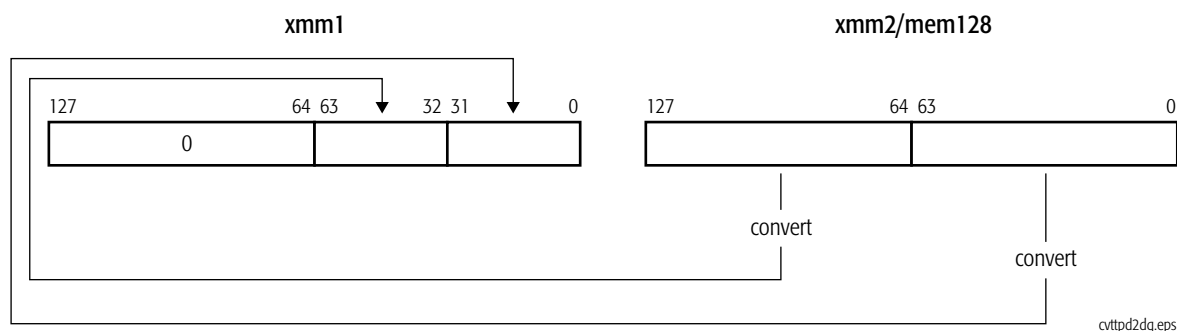
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTTPD2DQ**Convert Packed Double-Precision Floating-Point to Packed Doubleword Integers, Truncated**

The CVTTPD2DQ instruction converts two packed double-precision floating-point values in an XMM register or a 128-bit memory location to two packed 32-bit signed integer values and writes the converted values in the low-order 64 bits of another XMM register. The high-order 64 bits of the destination XMM register are cleared to all 0s.

Mnemonic	Opcode	Description
CVTTPD2DQ <i>xmm1, xmm2/mem128</i>	66 0F E6	Converts packed double-precision floating-point values in an XMM register or 128-bit memory location to packed doubleword integer values in the destination XMM register. Inexact results are truncated.



If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

	FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

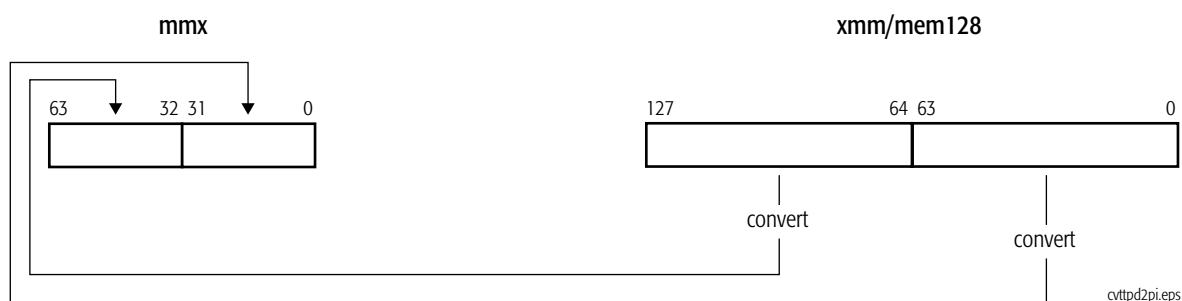
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTTPD2PI**Convert Packed Double-Precision Floating-Point to Packed Doubleword Integers, Truncated**

The CVTTPD2PI instruction converts two packed double-precision floating-point values in an XMM register or a 128-bit memory location to two packed 32-bit signed integer values and writes the converted values in an MMX register.

Mnemonic	Opcode	Description
CVTPD2PI <i>mmx, xmm/mem128</i>	66 0F 2C /r	Converts packed double-precision floating-point values in an XMM register or 128-bit memory location to packed doubleword integer values in the destination MMX™ register. Inexact results are truncated.



If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

Execution of this instruction causes all fields in the x87 tag word to be set according to their corresponding data, the top-of-stack-pointer bit (TOP) in the x87 status word to be cleared to 0, and any pending x87 exceptions are handled before this instruction is executed. For details, see “Actions Taken on Executing 64-Bit Media Instructions” in Volume 1.

Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

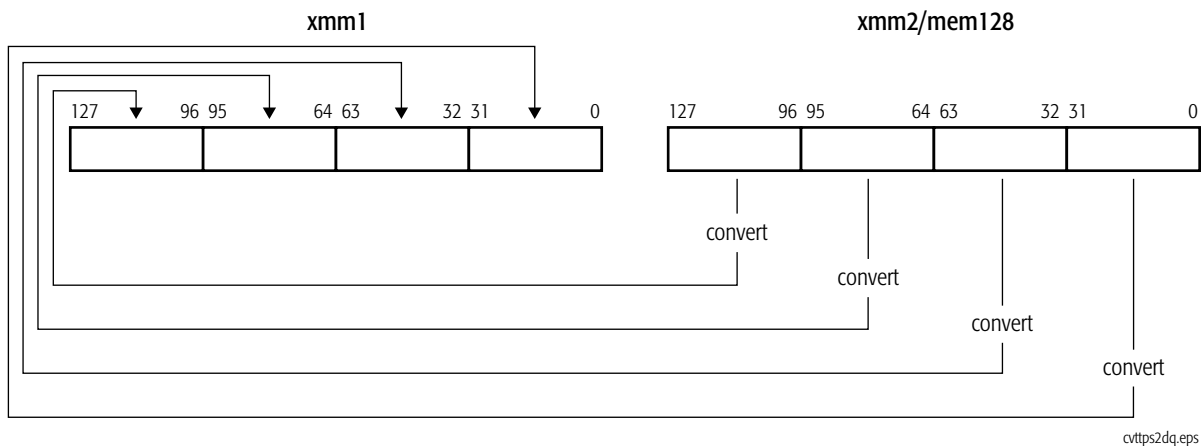
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception is pending due to an x87 floating-point instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTTPS2DQ**Convert Packed Single-Precision Floating-Point to Packed Doubleword Integers, Truncated**

The CVTTPS2DQ instruction converts four packed single-precision floating-point values in an XMM register or a 128-bit memory location to four packed 32-bit signed integers and writes the converted values in another XMM register.

Mnemonic	Opcode	Description
CVTTPS2DQ <i>xmm1, xmm2/mem128</i>	F3 0F 5B /r	Converts packed single-precision floating-point values in an XMM register or 128-bit memory location to packed doubleword integer values in the destination XMM register. Inexact results are truncated.



If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

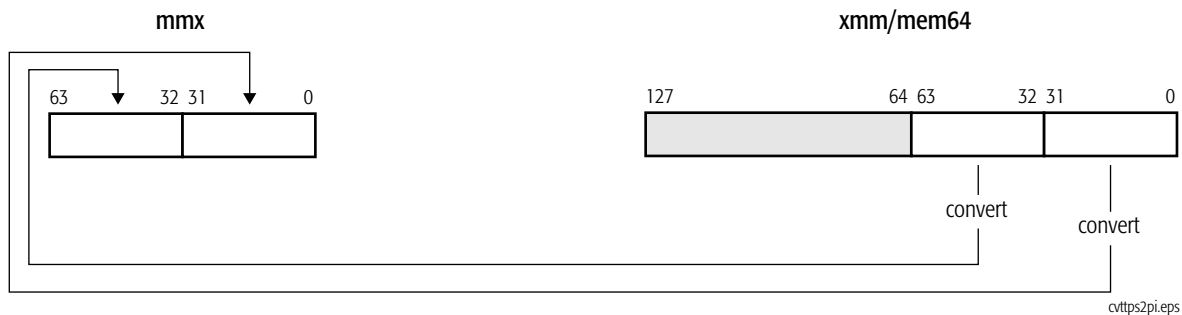
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTTPS2PI**Convert Packed Single-Precision Floating-Point to Packed Doubleword Integers, Truncated**

The CVTTPS2PI instruction converts two packed single-precision floating-point values in the low-order 64 bits of an XMM register or a 64-bit memory location to two packed 32-bit signed integer values and writes the converted values in an MMX register.

Mnemonic	Opcode	Description
CVTTPS2PI <i>mmx xmm/mem64</i>	0F 2C/r	Converts packed single-precision floating-point values in an XMM register or 64-bit memory location to doubleword integer values in the destination MMX™ register. Inexact results are truncated.



If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

Execution of this instruction causes all fields in the x87 tag word to be set according to their corresponding data, the top-of-stack-pointer bit (TOP) in the x87 status word to be cleared to 0, and any pending x87 exceptions are handled before this instruction is executed. For details, see “Actions Taken on Executing 64-Bit Media Instructions” in Volume 1.

Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.

Exceptions

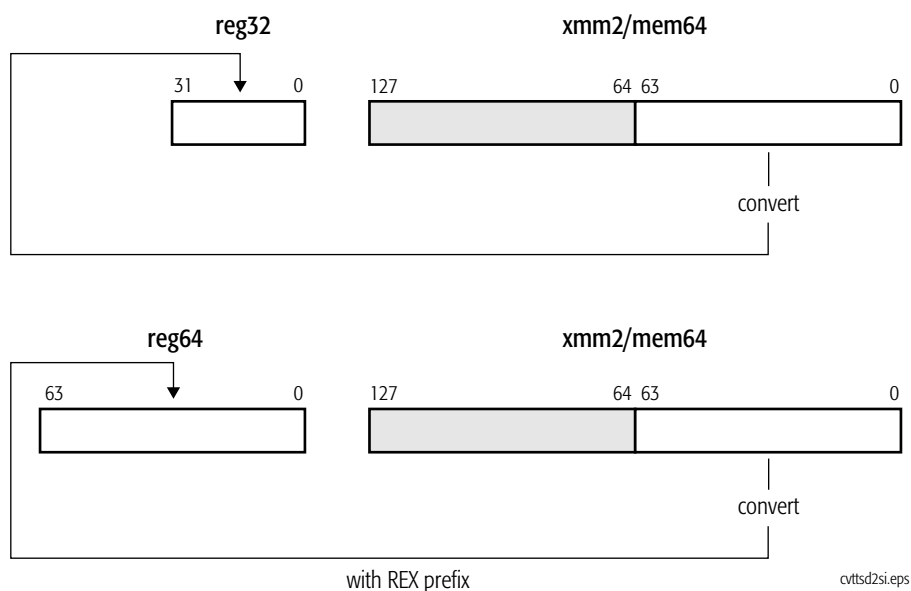
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception is pending due to an x87 floating-point instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTTSD2SI**Convert Scalar Double-Precision Floating-Point to Signed Doubleword of Quadword Integer, Truncated**

The CVTTSD2SI instruction converts a double-precision floating-point value in the low-order 64 bits of an XMM register or a 64-bit memory location to a 32-bit or 64-bit signed integer value and writes the converted value in a general-purpose register.

Mnemonic	Opcode	Description
CVTTSD2SI <i>reg32, xmm/mem64</i>	F2 0F 2C /r	Converts scalar double-precision floating-point value in an XMM register or 64-bit memory location to a doubleword signed integer value in a general-purpose register. Inexact results are truncated.
CVTTSD2SI <i>reg64, xmm/mem64</i>	F2 0F 2C /r	Converts scalar double-precision floating-point value in an XMM register or 64-bit memory location to a quadword signed integer value in a general-purpose register. Inexact results are truncated.



If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$) or quadword value (-2^{63} to $+2^{63} - 1$), the instruction returns the indefinite integer value

(8000_0000h for 32-bit integers, 8000_0000_0000_0000h for 64-bit integers) when the invalid-operation exception (IE) is masked.

Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Note: A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.																	

Exceptions

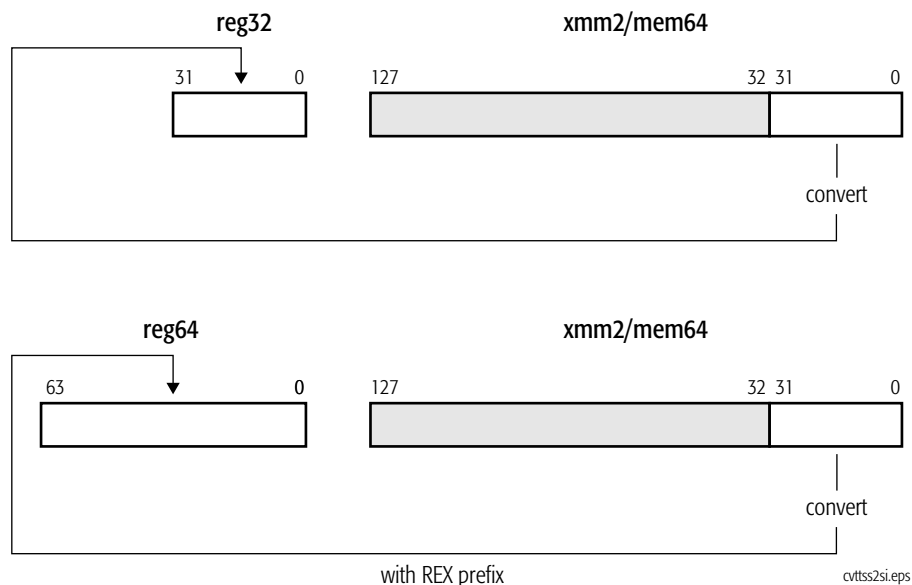
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

CVTTSS2SI**Convert Scalar Single-Precision Floating-Point to Signed Doubleword or Quadword Integer, Truncated**

The CVTTSS2SI instruction converts a single-precision floating-point value in the low-order 32 bits of an XMM register or a 32-bit memory location to a 32-bit or 64-bit signed integer value and writes the converted value in a general-purpose register.

Mnemonic	Opcode	Description
CVTTSS2SI <i>reg32, xmm/mem32</i>	F3 0F 2C /r	Converts scalar single-precision floating-point value in an XMM register or 32-bit memory location to a signed doubleword integer value in a general-purpose register. Inexact results are truncated.
CVTTSS2SI <i>reg64, xmm/mem32</i>	F3 0F 2C /r	Converts scalar single-precision floating-point value in an XMM register or 32-bit memory location to a signed quadword integer value in a general-purpose register. Inexact results are truncated.



If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$) or quadword value (-2^{63} to $+2^{63} - 1$), the instruction returns the indefinite integer value

(8000_0000h for 32-bit integers, 8000_0000_0000_0000h for 64-bit integers) when the invalid-operation exception (IE) is masked.

Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M					M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Note: A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.																	

Exceptions

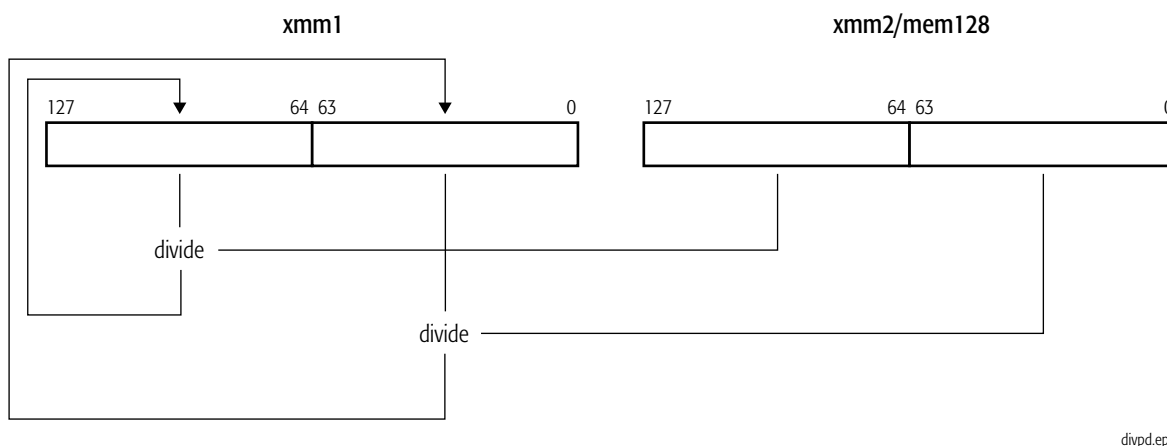
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN, QNaN, or infinity, or there is an overflow.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

DIVPD**Divide Packed Double-Precision Floating-Point**

The DIVPD instruction divides each of the two packed double-precision floating-point values in the first source operand by the corresponding packed double-precision floating-point value in the second source operand and writes the result of each division in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
DIVPD <i>xmm1, xmm2/mem128</i>	66 0F 5E/ <i>r</i>	Divides packed double-precision floating-point values in an XMM register by the packed double-precision floating-point values in another XMM register or 128-bit memory location.

**Related Instructions**

DIVPS, DIVSD, DIVSS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M	M	M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

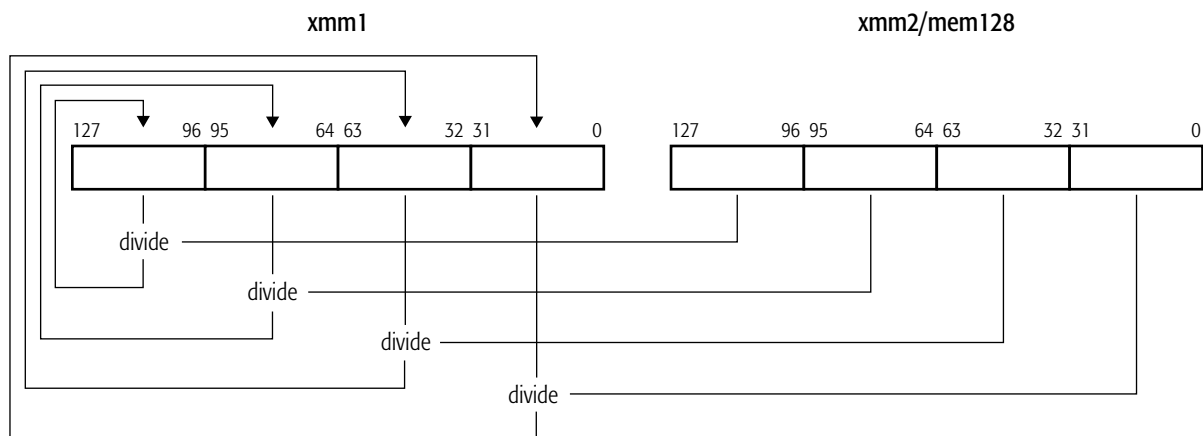
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Zero-divide exception (ZE)	X	X	X	Attempt to divide non-zero operand by zero.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

DIVPS**Divide Packed Single-Precision Floating-Point**

The DIVPS instruction divides each of the four packed single-precision floating-point values in the first source operand by the corresponding packed single-precision floating-point value in the second source operand and writes the result of each division in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
DIVPS <i>xmm1, xmm2mem/128</i>	OF 5E/ <i>r</i>	Divides packed single-precision floating-point values in an XMM register by the packed single-precision floating-point values in another XMM register or 128-bit memory location.

**Related Instructions**

DIVPD, DIVSD, DIVSS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M	M	M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

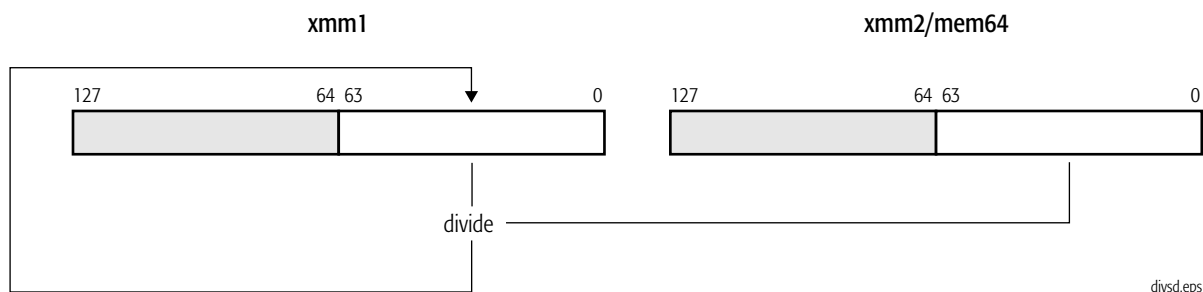
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Zero-divide exception (ZE)	X	X	X	Attempt to divide non-zero operand by zero.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

DIVSD**Divide Scalar Double-Precision Floating-Point**

The DIVSD instruction divides the double-precision floating-point value in the low-order quadword of the first source operand by the double-precision floating-point value in the low-order quadword of the second source operand and writes the result in the low-order quadword of the destination (first source). The high-order quadword of the destination is not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
DIVSD <i>xmm1, xmm2/mem64</i>	F2 0F 5E/ <i>r</i>	Divides low-order double-precision floating-point value in an XMM register by the low-order double-precision floating-point value in another XMM register or in a 64-bit memory location.

**Related Instructions**

DIVPD, DIVPS, DIVSS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M	M	M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

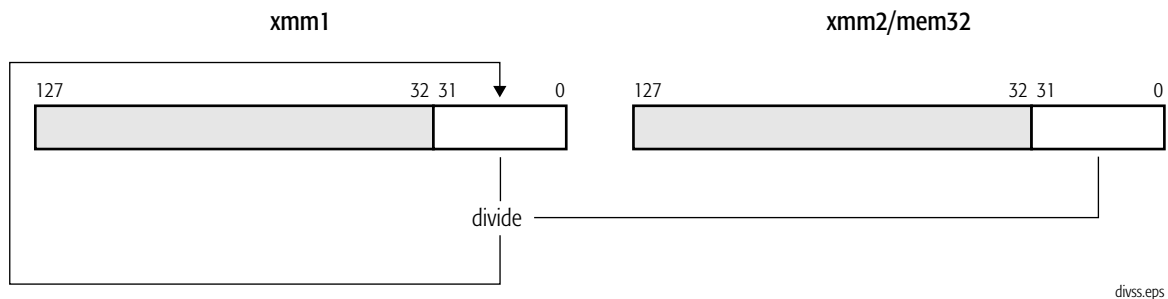
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Zero-divide exception (ZE)	X	X	X	Attempt to divide non-zero operand by zero.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

DIVSS**Divide Scalar Single-Precision Floating-Point**

The DIVSS instruction divides the single-precision floating-point value in the low-order doubleword of the first source operand by the single-precision floating-point value in the low-order doubleword of the second source operand and writes the result in the low-order doubleword of the destination (first source). The three high-order doublewords of the destination are not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
DIVSS <i>xmm1, xmm2/mem32</i>	F3 0F 5E/ <i>r</i>	Divides low-order single-precision floating-point value in an XMM register by the low-order single-precision floating-point value in another XMM register or in a 32-bit memory location.

**Related Instructions**

DIVPD, DIVPS, DIVSD

rFLAGS Affected

None

MXCSR Flags Affected.

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M	M	M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Zero-divide exception (ZE)	X	X	X	Attempt to divide non-zero operand by zero.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

FXRSTOR**Restore XMM, MMX™, and x87 State**

The FXRSTOR instruction restores the XMM, MMX, and x87 state. The data loaded from memory is the state information previously saved using the FXSAVE instruction. Restoring data with FXRSTOR that had been previously saved with an FSAVE (rather than FXSAVE) instruction results in an incorrect restoration.

Mnemonic	Opcode	Description
FXRSTOR <i>mem512env</i>	0F AE /1	Restores XMM, MMX™, and x87 state.

Unlike the FRSTOR instruction, FXRSTOR does not cause pending unmasked x87 floating-point exceptions in the restored image to be recognized when internal control of x87 exceptions is enabled (CR0.NE = 1). When loading a new environment, use an FWAIT instruction after the FXRSTOR instruction to check for and handle any pending unmasked x87 exceptions.

If the restored MXCSR register contains a set bit in an exception status flag, and the corresponding exception mask bit is cleared (indicating an unmasked exception), loading the MXCSR register from memory does not cause a SIMD floating-point exception (#XF).

FXRSTOR executes faster than FRSTOR because it does not restore the x87 error pointers (last instruction pointer, last data pointer, and last opcode), except in the relatively rare cases in which the exception-summary (ES) bit in the x87 status word is set to 1, indicating that an unmasked x87 exception has occurred.

The architecture supports two memory formats for FXRSTOR, a 512-byte 32-bit legacy format and a 512-byte 64-bit format. Selection of the 32-bit or 64-bit format is accomplished by using the corresponding effective operand size in the FXRSTOR instruction. If software running in 64-bit mode executes an FXRSTOR with a 32-bit operand size (no REX-prefix operand-size override), the 32-bit legacy format is used. If software running in 64-bit mode executes an FXRSTOR with a 64-bit operand size (requires REX-prefix operand-size override), the 64-bit format is used. For details about the memory image restored by FXRSTOR, see “Saving Media and x87 Processor State” in Volume 2.

If the operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0, the saved image of XMM0–XMM7 and MXCSR is not loaded into the processor. A general-protection exception occurs if there is an attempt to load a non-zero value to the bits in MXCSR that are defined as reserved and cleared (bits 31–16 and bit 6).

Related Instructions

FWAIT, FXSAVE

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	Instruction preceded by LOCK prefix. The emulate bit (EM) of CR0 is set to 1. The FXSAVE/FSRSTOR bit (bit 24) in CPUID standard function 1 or extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS). An attempt is made to write a non-zero value to reserved bits in MXCSR.
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	If alignment checks are enabled (i.e., the AC bit of rFLAGS is set to 1, the AM bit of CR0 are set to 1, and in protected mode, CPL = 3), reporting of #AC vs. #GP is a function of the misalignment and is implementation-dependent. If alignment checks are not enabled, only 16-byte misalignment is reported via #GP.

FXSAVE Save XMM, MMX™, and x87 State

The FXSAVE instruction saves the XMM, MMX, and x87 state. A memory location that is not aligned on a 16-byte boundary causes a general-protection exception or, in some cases, an alignment-check exception.

Mnemonic	Opcode	Description
FXSAVE <i>mem512env</i>	OF AE /0	Saves XMM, MMX™, and x87 state in 512-byte memory location.

Unlike FSAVE and FNSAVE, FXSAVE does not alter the x87 tag bits. The contents of the saved MMX/x87 data registers are retained, thus indicating that the registers may be valid (or whatever other value the x87 tag bits indicated prior to the save). To invalidate the contents of the MMX/x87 data registers after FXSAVE, software must execute an FINIT instruction. Also, FXSAVE (like FNSAVE) does not check for pending unmasked x87 floating-point exceptions. An FWAIT instruction can be used for this purpose.

FXSAVE executes faster than FSAVE or FNSAVE because it does not save the x87 pointer registers (last instruction pointer, last data pointer, and last opcode), except in the relatively rare cases in which the exception-summary (ES) bit in the x87 status word is set to 1, indicating that an unmasked x87 exception has occurred.

The architecture supports two memory formats for FXSAVE, a 512-byte 32-bit legacy format and a 512-byte 64-bit format. Selection of the 32-bit or 64-bit format is accomplished by using the corresponding effective operand size in the FXSAVE instruction. If software running in 64-bit mode executes an FXSAVE with a 32-bit operand size (no REX-prefix operand-size override), the 32-bit legacy format is used. If software running in 64-bit mode executes an FXSAVE with a 64-bit operand size (requires REX-prefix operand-size override), the 64-bit format is used. For details about the memory image restored by FXRSTOR, see “Saving Media and x87 Processor State” in Volume 2.

If the operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0, FXSAVE does not save the image of XMM0–XMM7 and MXCSR. For details about the CR4.OSFXSR bit, see “FXSAVE/FXRSTOR Support (OSFXSR) Bit” in Volume 2.

Related Instructions

FINIT, FNSAVE, FRSTOR, FSAVE, FXRSTOR, LDMXCSR, STMXCSR

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	Instruction preceded by LOCK prefix. The emulate bit (EM) of CR0 is set to 1. The FXSAVE/FSRSTOR bit (bit 24) in CPUID standard function 1 or extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS). An attempt is made to write a non-zero value to reserved bits in MXCSR.
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	If alignment checks are enabled (i.e., the AC bit of rFLAGS is set to 1, the AM bit of CR0 are set to 1, and in protected mode, CPL = 3), reporting of #AC vs. #GP is a function of the misalignment and is implementation-dependent. If alignment checks are not enabled, only 16-byte misalignment is reported via #GP.

LDMXCSR**Load MXCSR Control/Status Register**

The LDMXCSR instruction loads the MXCSR register with a 32-bit value from memory. The least-significant bit of the memory location is loaded in bit 0 of MXCSR. Bits 31–16 and bit 6 of the MXCSR are reserved and must be zero. A general-protection exception occurs if the LDMXCSR instruction attempts to load non-zero values into MXCSR bits 31–16 or bit 6.

The MXCSR register is described in “Registers” in Volume 1.

MnemonicLDMXCSR *mem32***Opcode**

OF AE/2

Description

Loads MXCSR register with 32-bit value in memory.

Related Instructions

STMXCSR

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

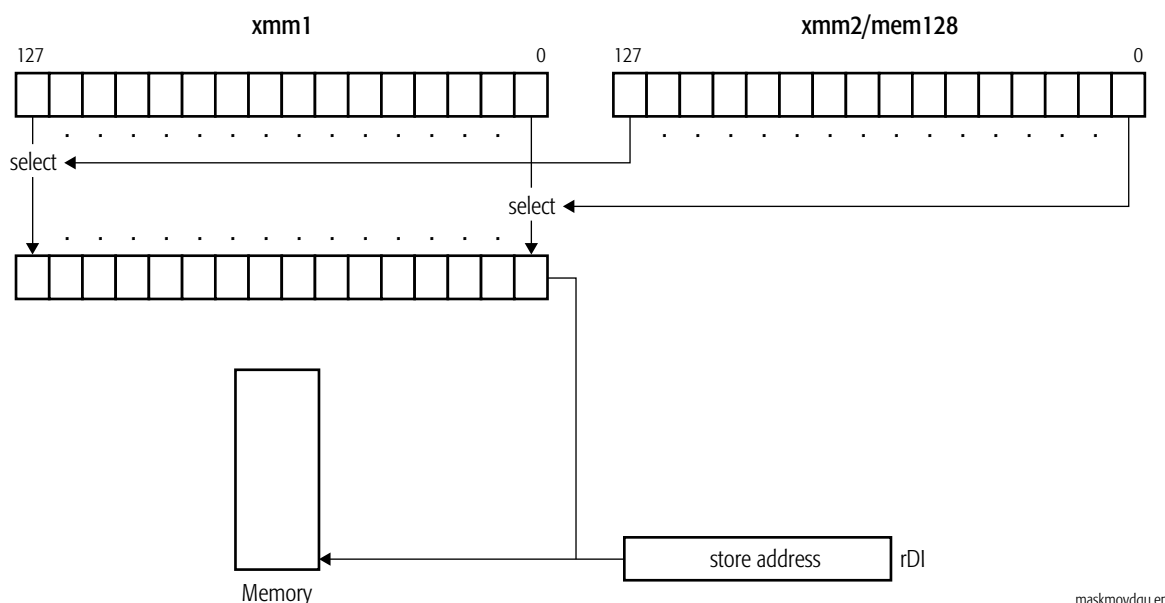
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)

Exception	Real	Virtual 8086	Protected	Cause of Exception
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

MASKMOVDQU Masked Move Double Quadword Unaligned

The MASKMOVDQU instruction stores bytes from the first source operand as selected by the mask value in the second source operand (0 = no write and 1 = write) to a memory location specified in the rDI and DS registers. The first source operand is an XMM register, and the second source operand is another XMM register. The store address may be unaligned.

Mnemonic	Opcode	Description
MASKMOVDQU <i>xmm1, xmm2</i>	66 0F F7 /r	Store bytes from an XMM register selected by a mask value in another XMM register to a memory location.



maskmovdqu.eps

A mask value of all 0s results in the following behavior:

- No data is written to memory.
- Code and data breakpoints are not guaranteed to be signaled in all implementations.
- Exceptions associated with memory addressing and page faults are not guaranteed to be signaled in all implementations.
- The protection features of memory regions mapped as UC or WP are not guaranteed to be enforced in all implementations.

MASKMOVDQU implicitly uses weakly-ordered, write-combining buffering for the data, as described in “Buffering and Combining Memory Writes” in Volume 2. For data that is shared by multiple processors, this instruction should be used together with a fence instruction in order to ensure data coherency (refer to “Cache and TLB Management” in Volume 2).

Related Instructions

MASKMOVQ

rFLAGS Affected

None

MXCSR Flags Affected

None

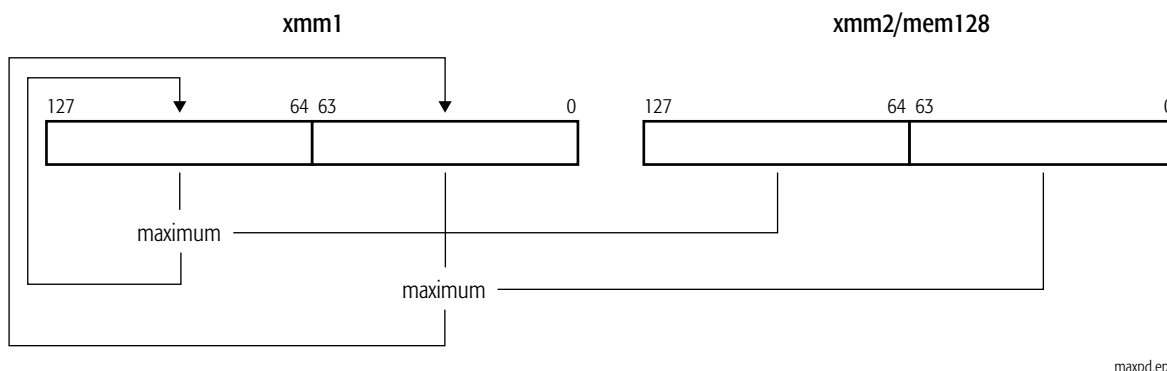
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

MAXPD**Maximum Packed Double-Precision Floating-Point**

The MAXPD instruction compares each of the two packed double-precision floating-point values in the first source operand with the corresponding packed double-precision floating-point value in the second source operand and writes the numerically greater of the two values for each comparison in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
MAXPD <i>xmm1, xmm2/mem128</i>	66 0F 5F /r	Compares two pairs of packed double-precision values in an XMM register and another XMM register or 128-bit memory location and writes the greater value of each comparison in the destination XMM register.



If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), the second source operand is written to the destination.

Related Instructions

MAXPS, MAXSD, MAXSS, MINPD, MINPS, MINS, MINSS

rFLAGS Affected

None

MXCSR Flags Affected

	FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

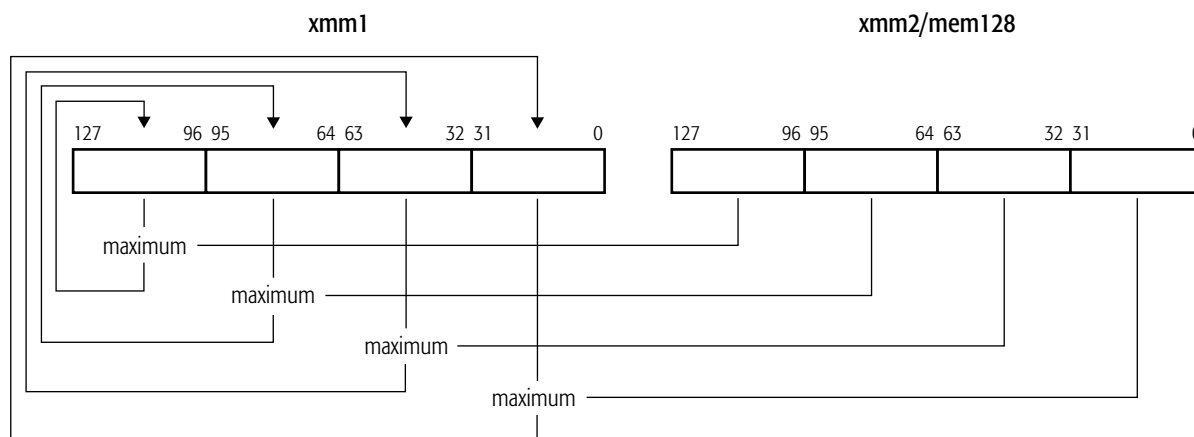
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

MAXPS**Maximum Packed Single-Precision Floating-Point**

The MAXPS instruction compares each of the four packed single-precision floating-point values in the first source operand with the corresponding packed single-precision floating-point value in the second source operand and writes the numerically greater of the two values for each comparison in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
MAXPS <i>xmm1, xmm2/mem128</i>	0F 5F /r	Compares four pairs of packed single-precision values in an XMM register and another XMM register or 128-bit memory location and writes the maximum value of each comparison in the destination XMM register.



maxps.eps

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), the second source operand is written to the destination.

Related Instructions

MAXPD, MAXSD, MAXSS, MINPD, MINPS, MINSR, MINSS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

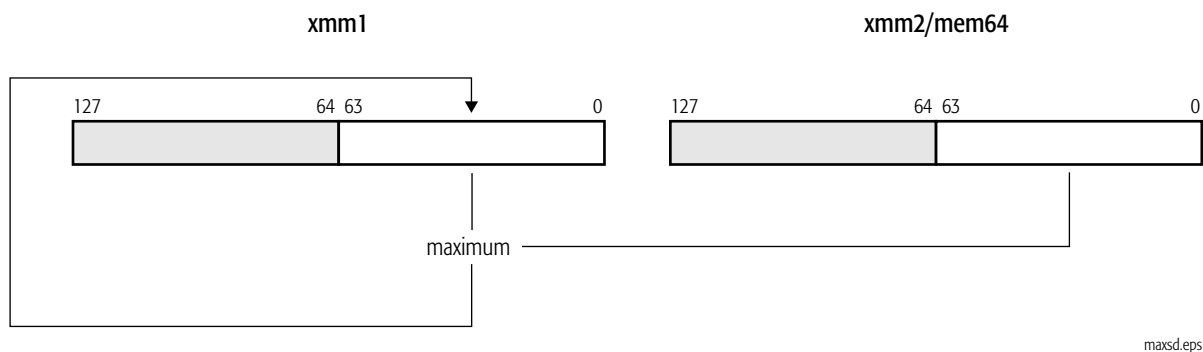
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

MAXSD**Maximum Scalar Double-Precision Floating-Point**

The MAXSD instruction compares the double-precision floating-point value in the low-order 64 bits of the first source operand with the double-precision floating-point value in the low-order 64 bits of the second source operand and writes the numerically greater of the two values in the low-order quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 64-bit memory location. The high-order quadword of the destination XMM register is not modified.

Mnemonic	Opcode	Description
MAXSD <i>xmm1, xmm2/mem64</i>	F2 0F 5F /r	Compares scalar double-precision values in an XMM register and another XMM register or 64-bit memory location and writes the greater of the two values in the destination XMM register.



maxsd.eps

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), the second source operand is written to the destination.

Related Instructions

MAXPD, MAXPS, MAXSS, MINPD, MINPS, MINSD, MINSS

rFLAGS Affected

None

MXCSR Flags Affected

	FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

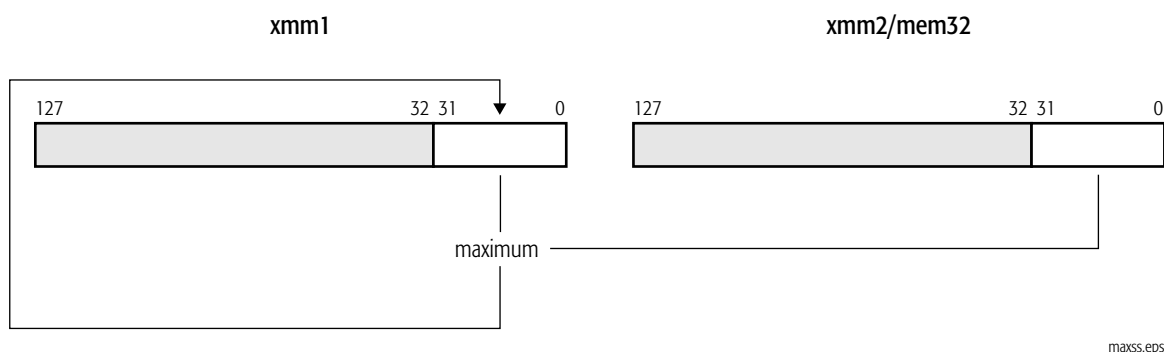
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

MAXSS Maximum Scalar Single-Precision Floating-Point

The MAXSS instruction compares the single-precision floating-point value in the low-order 32 bits of the first source operand with the single-precision floating-point value in the low-order 32 bits of the second source operand and writes the numerically greater of the two values in the low-order 32 bits of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 32-bit memory location. The three high-order doublewords of the destination XMM register are not modified.

Mnemonic	Opcode	Description
MAXSS <i>xmm1, xmm2/mem32</i>	F3 0F 5F /r	Compares scalar single-precision floating-point values in an XMM register and another XMM register or 32-bit memory location and writes the greater of the two values in the destination XMM register.



If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), the second source operand is written to the destination.

Related Instructions

MAXPD, MAXPS, MAXSD, MINPD, MINPS, MINSD, MINSS, PFMAX

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																	M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

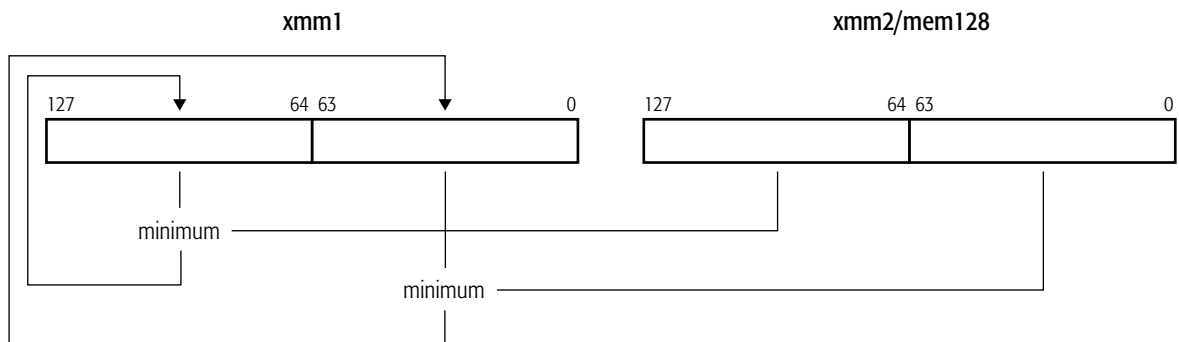
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

MINPD Minimum Packed Double-Precision Floating-Point

The MINPD instruction compares each of the two packed double-precision floating-point values in the first source operand with the corresponding packed double-precision floating-point value in the second source operand and writes the numerically lesser of the two values for each comparison in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 128-bit memory location.

Mnemonic	Opcode	Description
MINPD <i>xmm1, xmm2/mem128</i>	66 0F 5D /r	Compares two pairs of packed double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the lesser value of each comparison in the destination XMM register.



minpd.eps

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), the second source operand is written to the destination.

Related Instructions

MAXPD, MAXPS, MAXSD, MAXSS, MINPS, MINSD, MINSS

rFLAGS Affected

None

MXCSR Flags Affected

	FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

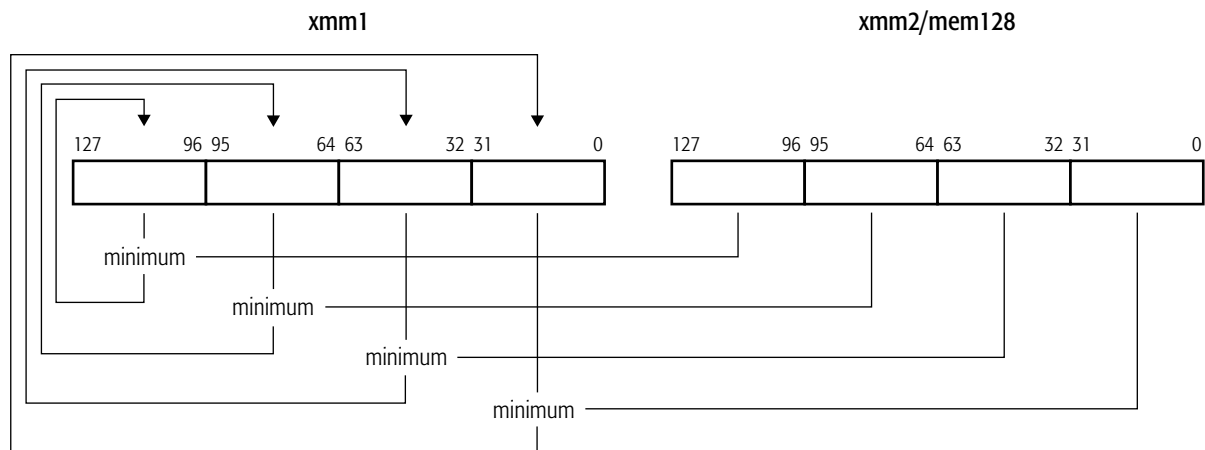
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

MINPS**Minimum Packed Single-Precision Floating-Point**

The MINPS instruction compares each of the four packed single-precision floating-point values in the first source operand with the corresponding packed single-precision floating-point value in the second source operand and writes the numerically lesser of the two values for each comparison in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 128-bit memory location.

Mnemonic	Opcode	Description
MINPS <i>xmm1, xmm2/mem128</i>	OF 5D /r	Compares four pairs of packed single-precision values in an XMM register and another XMM register or 128-bit memory location and writes the numerically lesser value of each comparison in the destination XMM register.



minps.eps

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), the second source operand is written to the destination.

Related Instructions

MAXPD, MAXPS, MAXSD, MAXSS, MINPD, MINSD, MINSS, PFMIN

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

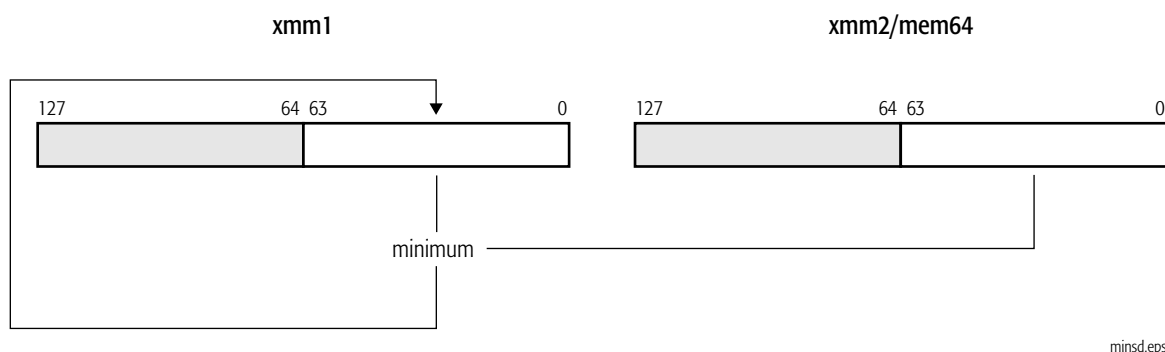
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

MINSD **Minimum Scalar Double-Precision Floating-Point**

The MINSD instruction compares the double-precision floating-point value in the low-order 64 bits of the first source operand with the double-precision floating-point value in the low-order 64 bits of the second source operand and writes the numerically lesser of the two values in the low-order 64 bits of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 64-bit memory location. The high-order quadword of the destination XMM register is not modified.

Mnemonic	Opcode	Description
MINSD <i>xmm1, xmm2/mem64</i>	F2 0F 5D /r	Compares scalar double-precision floating-point values in an XMM register and another XMM register or 64-bit memory location and writes the lesser of the two values in the destination XMM register.



If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), the second source operand is written to the destination.

Related Instructions

MAXPD, MAXPS, MAXSD, MAXSS, MINPD, MINPS, MINSS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																	M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

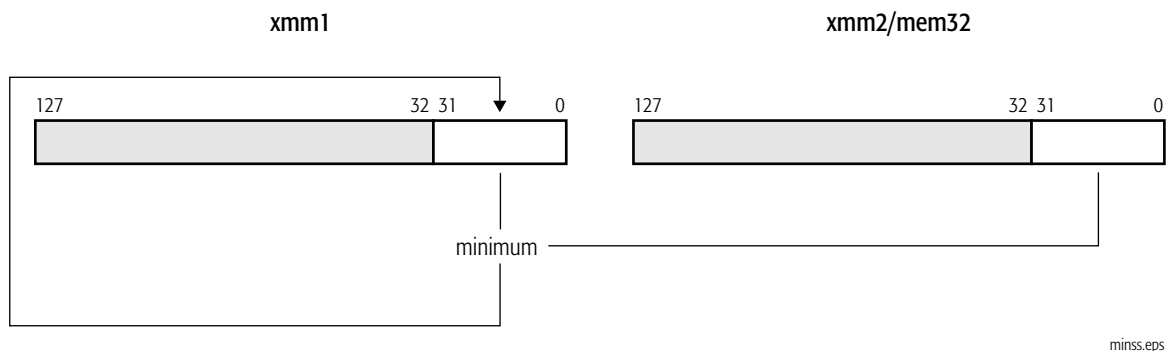
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

MINSS**Minimum Scalar Single-Precision Floating-Point**

The MINSD instruction compares the single-precision floating-point value in the low-order 32 bits of the first source operand with the single-precision floating-point value in the low-order 32 bits of the second source operand and writes the numerically lesser of the two values in the low-order 32 bits of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 32-bit memory location. The three high-order doublewords of the destination XMM register are not modified.

Mnemonic	Opcode	Description
MINSS <i>xmm1, xmm2/mem32</i>	F3 0F 5D /r	Compares scalar single-precision floating-point values in an XMM register and another XMM register or 32-bit memory location and writes the lesser of the two values in the destination XMM register.



If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), the second source operand is written to the destination.

Related Instructions

MAXPD, MAXPS, MAXSD, MAXSS, MINPD, MINPS, MINSD

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																	M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

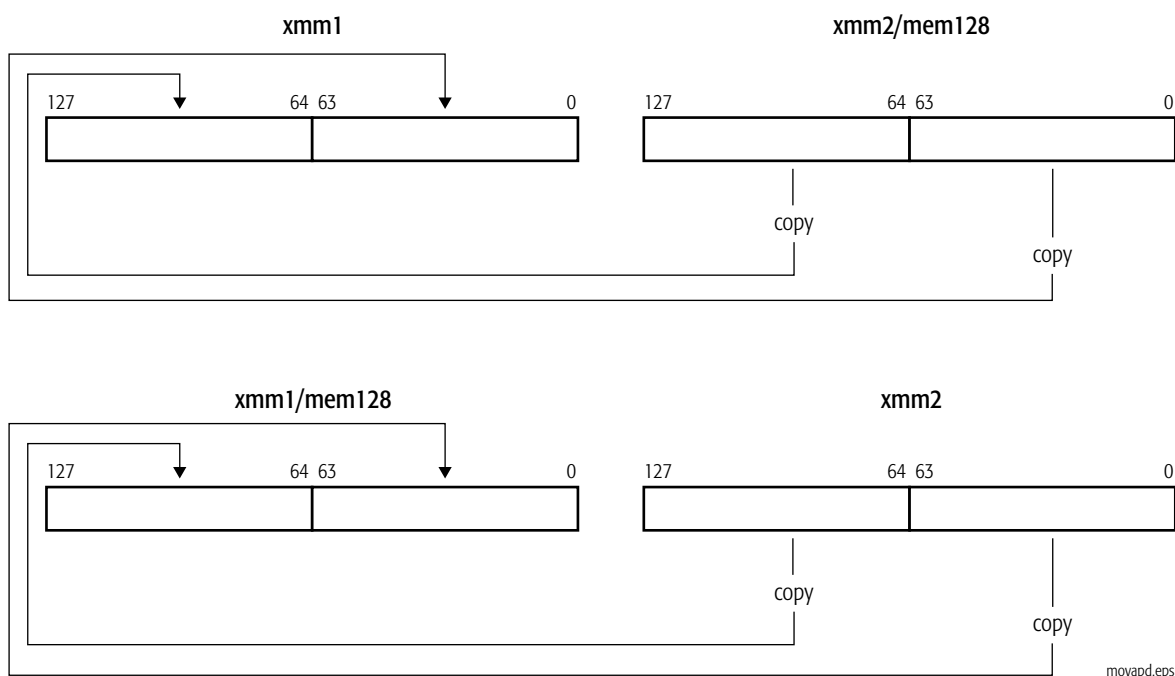
Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

MOVAPD**Move Aligned Packed Double-Precision Floating-Point**

The MOVAPD instruction moves two packed double-precision floating-point values:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
MOVAPD <i>xmm1, xmm2/mem128</i>	66 0F 28 /r	Moves packed double-precision floating-point value from an XMM register or 128-bit memory location to an XMM register.
MOVAPD <i>xmm1/mem128, xmm2</i>	66 0F 29 /r	Moves packed double-precision floating-point value from an XMM register to an XMM register or 128-bit memory location.



A memory operand that is not aligned on a 16-byte boundary causes a general-protection exception.

Related Instructions

MOVHPD, MOVLPD, MOVMSKPD, MOVSD, MOVUPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

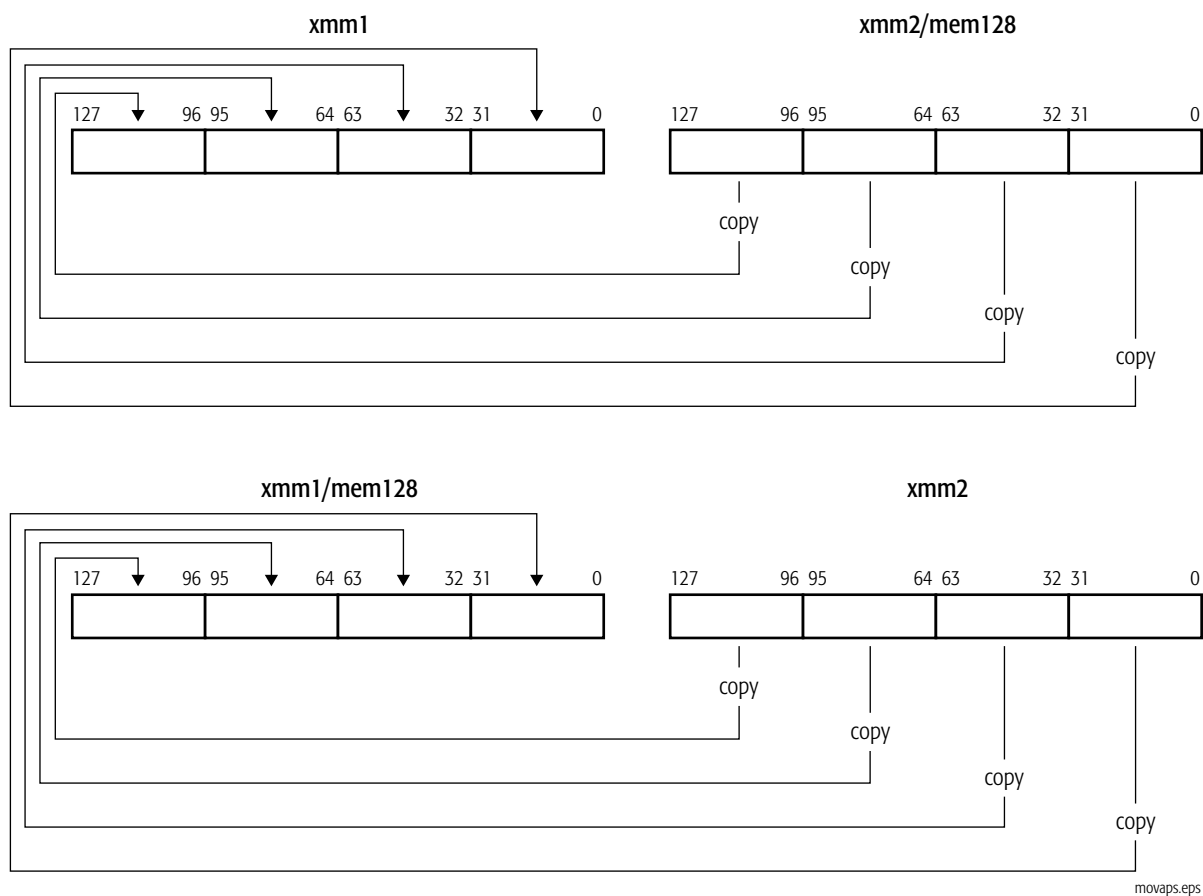
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

MOVAPS Move Aligned Packed Single-Precision Floating-Point

The MOVAPS instruction moves four packed single-precision floating-point values:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
MOVAPS <i>xmm1, xmm2/mem128</i>	0F 28 /r	Moves aligned packed single-precision floating-point value from an XMM register or 128-bit memory location to the destination XMM register.
MOVAPS <i>xmm1/mem128, xmm2</i>	0F 29 /r	Moves aligned packed single-precision floating-point value from an XMM register to the destination XMM register or 128-bit memory location.



A memory operand that is not aligned on a 16-byte boundary causes a general-protection exception.

Related Instructions

MOVHLPS, MOVHPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVSS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

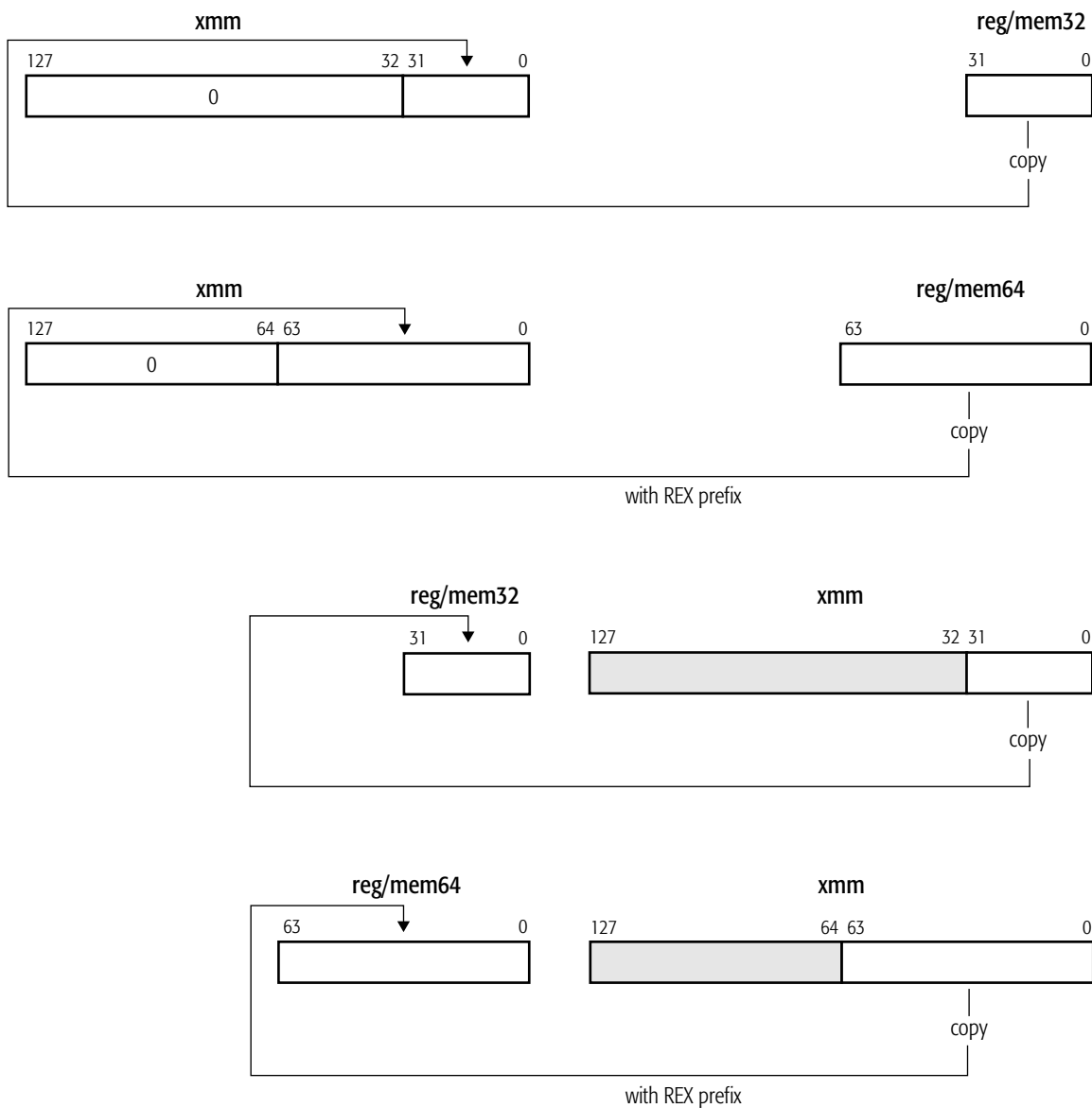
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

MOVD**Move Doubleword or Quadword**

The MOVD instruction moves a 32-bit or 64-bit value in one of the following ways:

- from a 32-bit or 64-bit general-purpose register or memory location to the low-order 32 or 64 bits of an XMM register, with zero-extension to 128 bits
- from the low-order 32 or 64 bits of an XMM to a 32-bit or 64-bit general-purpose register or memory location

Mnemonic	Opcode	Description
MOVD <i>xmm, reg/mem32</i>	66 0F 6E /r	Moves 32-bit value from a general-purpose register or 32-bit memory location to an XMM register.
MOVD <i>xmm, reg/mem64</i>	66 0F 6E /r	Moves 64-bit value from a general-purpose register or 64-bit memory location to an XMM register.
MOVD <i>reg/mem32, xmm</i>	66 0F 7E /r	Moves 32-bit value from an XMM register to a general-purpose register or 32-bit memory location.
MOVD <i>reg/mem64, xmm</i>	66 0F 7E /r	Moves 64-bit value from an XMM register to a general-purpose register or 64-bit memory location.



movd-128.eps

Related Instructions

MOVDQA, MOVDQU, MOVDQ2Q, MOVQ, MOVQ2DQ

rFLAGS Affected

None

MXCSR Flags Affected

None

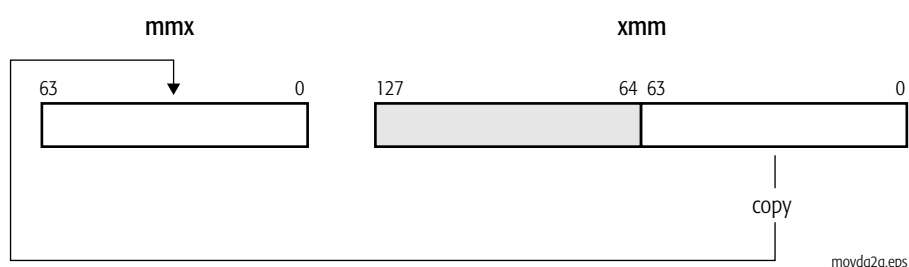
Exceptions

Exception	Real	Virtual 8086	Protected	Description
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

MOVDQ2Q**Move Quadword to Quadword**

The MOVDQ2Q instruction moves the low-order 64-bit value in an XMM register to a 64-bit MMX register.

Mnemonic	Opcode	Description
MOVDQ2Q <i>mmx, xmm</i>	F2 0F D6	Moves low-order 64-bit value from an XMM register to the destination MMX™ register.



Execution of this instruction causes all fields in the x87 tag word to be set according to their corresponding data, the top-of-stack-pointer bit (TOP) in the x87 status word to be cleared to 0, and any pending x87 exceptions are handled before this instruction is executed. For details, see “Actions Taken on Executing 64-Bit Media Instructions” in Volume 1.

Related Instructions

MOVD, MOVDQA, MOVDQU, MOVQ, MOVQ2DQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

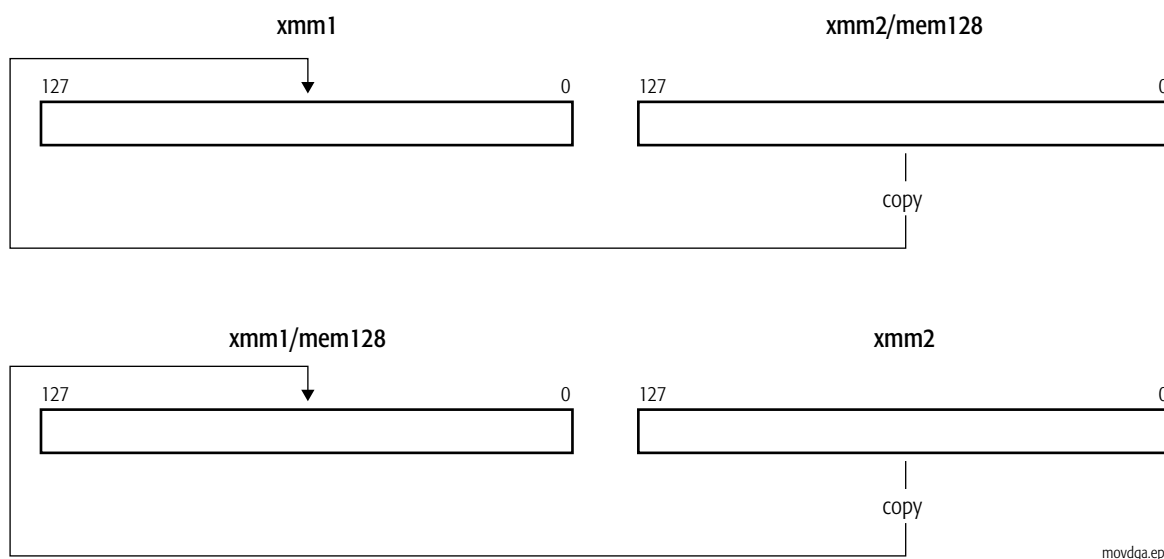
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
x87 floating-point exception pending, #MF	X	X	X	An exception is pending due to an x87 floating-point instruction.

MOVDQA**Move Aligned Double Quadword**

The MOVDQA instruction moves an aligned 128-bit (double quadword) value:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to a 128-bit memory location or another XMM register.

Mnemonic	Opcode	Description
MOVDQA <i>xmm1, xmm2/mem128</i>	66 0F 6F /r	Moves 128-bit value from an XMM register or 128-bit memory location to the destination XMM register.
MOVDQA <i>xmm1/mem128, xmm2</i>	66 0F 7F /r	Moves 128-bit value from an XMM register to the destination XMM register or 128-bit memory location.



A memory operand that is not aligned on a 16-byte boundary causes a general-protection exception.

Related Instructions

MOVD, MOVDQU, MOVDQ2Q, MOVQ, MOVQ2DQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

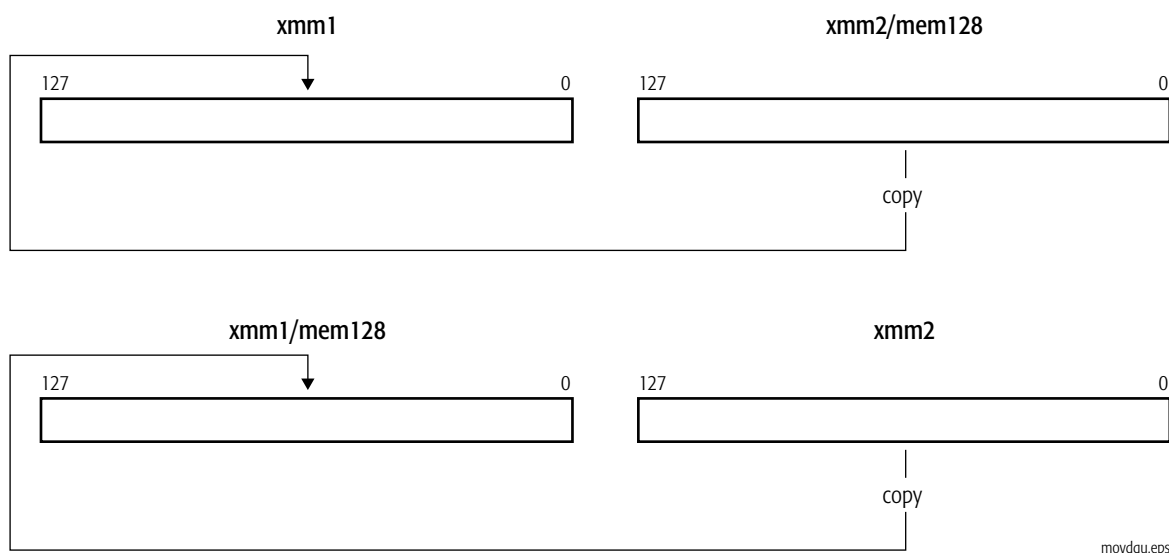
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

MOVDQU**Move Unaligned Double Quadword**

The MOVDQU instruction moves an unaligned 128-bit (double quadword) value:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
MOVDQU <i>xmm1</i> , <i>xmm2/mem128</i>	F3 0F 6F /r	Moves unaligned 128-bit value from an XMM register or 128-bit memory location to the destination XMM register.
MOVDQU <i>xmm1/mem128</i> , <i>xmm2</i>	F3 0F 7F /r	Moves unaligned 128-bit value from an XMM register to the destination XMM register or 128-bit memory location.



movdqu.eps

Memory operands that are not aligned on a 16-byte boundary do not cause a general-protection exception.

Related Instructions

MOVD, MOVDQA, MOVDQ2Q, MOVQ, MOVQ2DQ

rFLAGS Affected

None

MXCSR Flags Affected

None

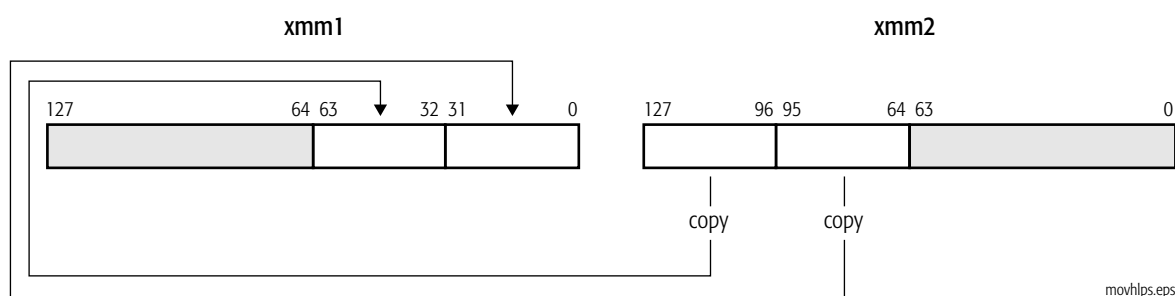
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

MOVHLPS Move Packed Single-Precision Floating-Point High to Low

The MOVHLPS instruction moves two packed single-precision floating-point values from the high-order 64 bits of an XMM register to the low-order 64 bits of another XMM register. The high-order 64 bits of the destination XMM register are not modified.

Mnemonic	Opcode	Description
MOVHLPS <i>xmm1</i> , <i>xmm2</i>	0F 12 /r	Moves two packed single-precision floating-point values from an XMM register to the destination XMM register.



Related Instructions

MOVAPS, MOVHPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVSS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.

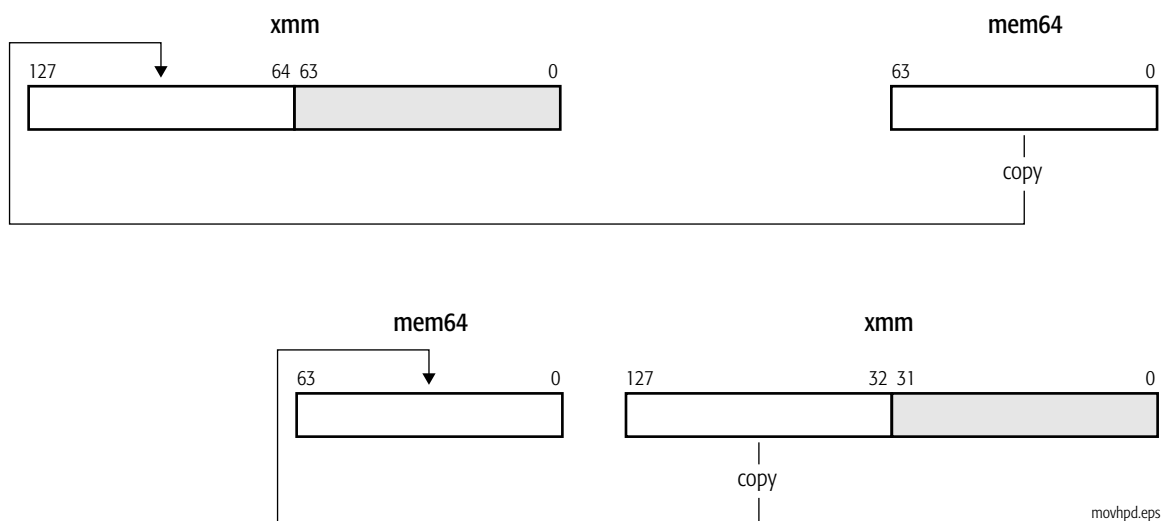
MOVHPD Move High Packed Double-Precision Floating-Point

The MOVHPD instruction moves a double-precision floating-point value:

- from a 64-bit memory location to the high-order 64 bits of an XMM register, or
- from the high-order 64 bits of an XMM register to a 64-bit memory location.

The low-order 64 bits of the destination XMM register are not modified.

Mnemonic	Opcode	Description
MOVHPD <i>xmm</i> , <i>mem64</i>	66 0F 16 /r	Moves double-precision floating-point value from a 64-bit memory location to an XMM register.
MOVHPD <i>mem64</i> , <i>xmm</i>	66 0F 17 /r	Moves double-precision floating-point value from an XMM register to a 64-bit memory location.



Related Instructions

MOVAPD, MOVLPD, MOVMSKPD, MOVSD, MOVUPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

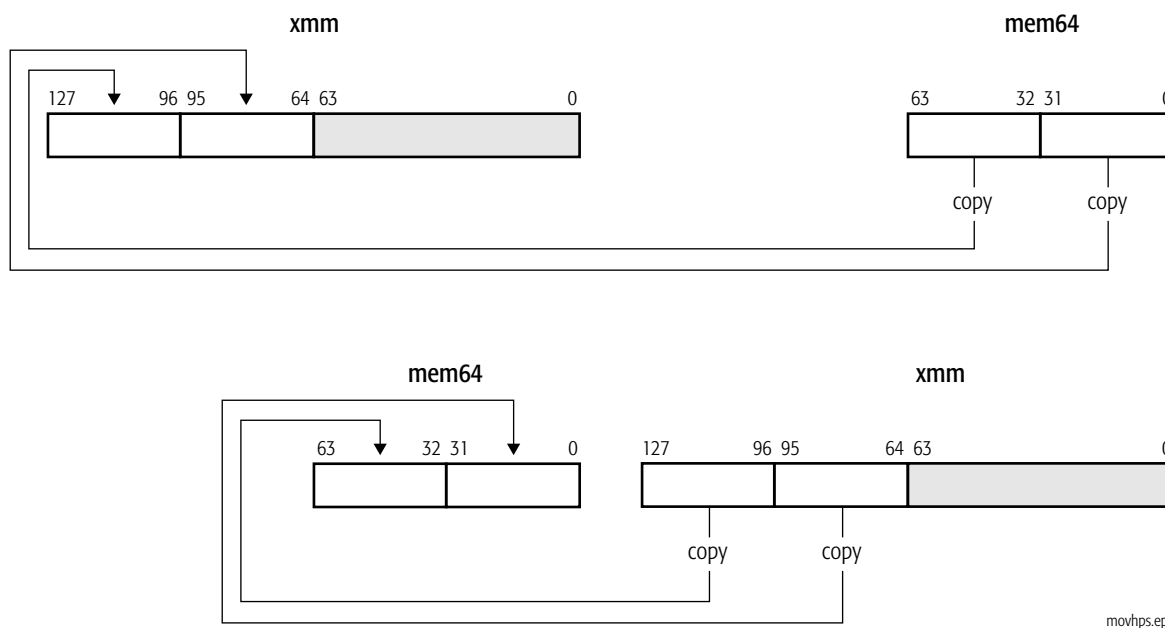
MOVHPS Move High Packed Single-Precision Floating-Point

The MOVHPS instruction moves two packed single-precision floating-point values:

- from a 64-bit memory location to the high-order 64 bits of an XMM register, or
- from the high-order 64 bits of an XMM register to a 64-bit memory location.

The low-order 64 bits of the destination XMM register are not modified.

Mnemonic	Opcode	Description
MOVHPS <i>xmm</i> , <i>mem64</i>	0F 16 /r	Moves two packed single-precision floating-point values from a 64-bit memory location to an XMM register.
MOVHPS <i>mem64</i> , <i>xmm</i>	0F 17 /r	Moves two packed single-precision floating-point values from an XMM register to a 64-bit memory location.



Related Instructions

MOVAPS, MOVHLPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVSS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

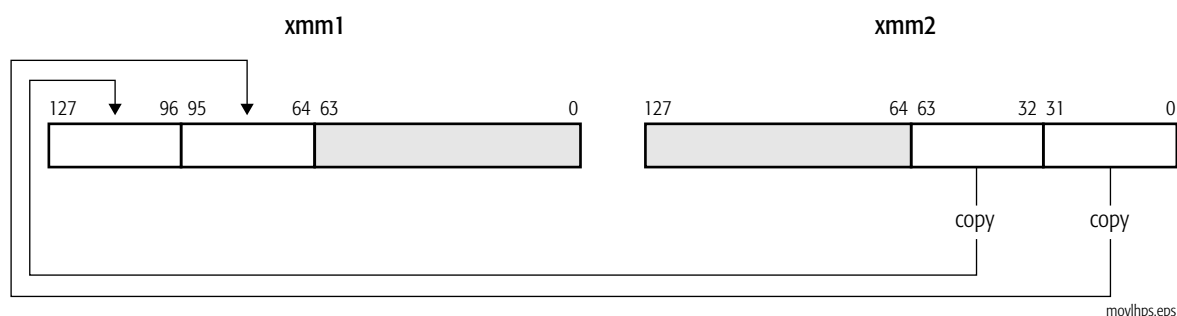
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

MOVLHPS**Move Packed Single-Precision Floating-Point
Low to High**

The MOVLHPS instruction moves two packed single-precision floating-point values from the low-order 64 bits of an XMM register to the high-order 64 bits of another XMM register. The low-order 64 bits of the destination XMM register are not modified.

Mnemonic	Opcode	Description
MOVLHPS <i>xmm1, xmm2</i>	OF 16 /r	Moves two packed single-precision floating-point values from an XMM register to another XMM register.

**Related Instructions**

MOVAPS, MOVHLPS, MOVHPS, MOVLPS, MOVMSKPS, MOVSS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.

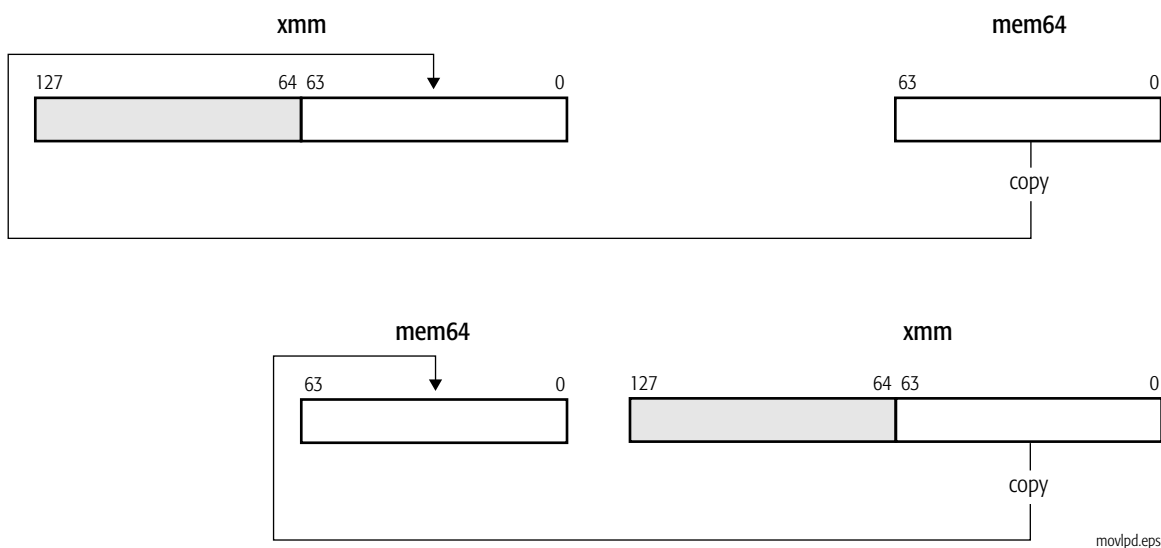
MOVLPD Move Low Packed Double-Precision Floating-Point

The MOVLPD instruction moves a double-precision floating-point value:

- from a 64-bit memory location to the low-order 64 bits of an XMM register, or
- from the low-order 64 bits of an XMM register to a 64-bit memory location.

The high-order 64 bits of the destination XMM register are not modified.

Mnemonic	Opcode	Description
MOVLPD <i>xmm, mem64</i>	66 0F 12 /r	Moves double-precision floating-point value from a 64-bit memory location to an XMM register.
MOVLPD <i>mem64, xmm</i>	66 0F 13 /r	Moves double-precision floating-point value from an XMM register to a 64-bit memory location.



Related Instructions

MOVAPD, MOVHPD, MOVMSKPD, MOVSD, MOVUPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

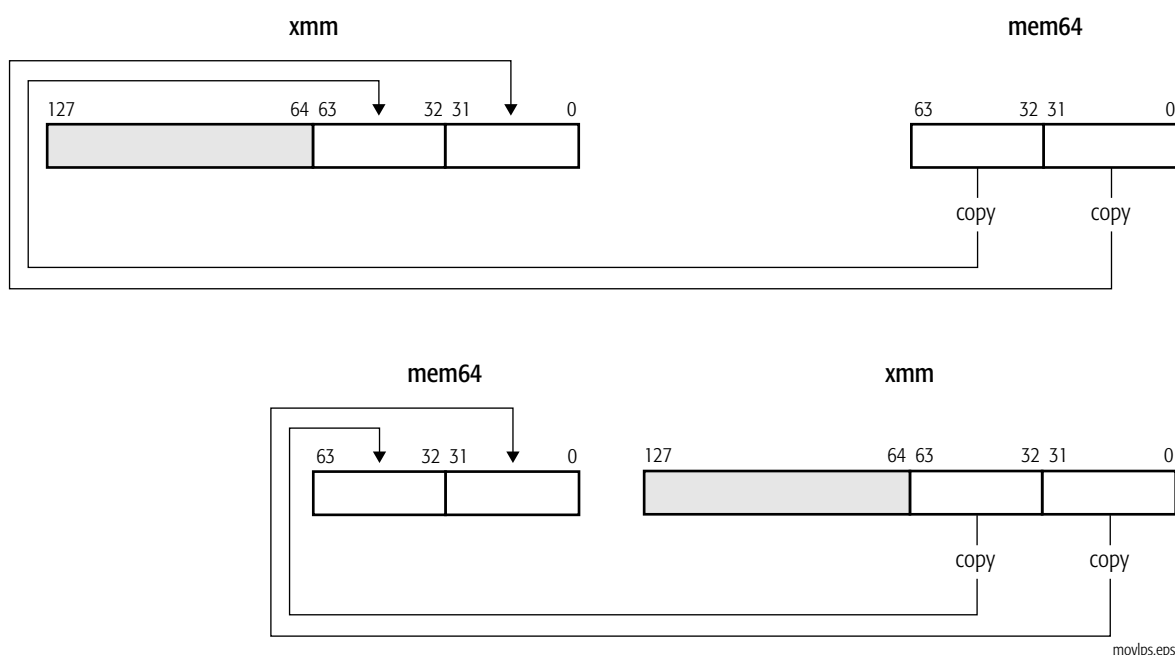
MOVLPS Move Low Packed Single-Precision Floating-Point

The MOVLPS instruction moves two packed single-precision floating-point values:

- from a 64-bit memory location to the low-order 64 bits of an XMM register, or
- from the low-order 64 bits of an XMM register to a 64-bit memory location

The high-order 64 bits of the destination XMM register are not modified.

Mnemonic	Opcode	Description
MOVLPS <i>xmm, mem64</i>	0F 12/ <i>r</i>	Moves two packed single-precision floating-point values from a 64-bit memory location to an XMM register.
MOVLPS <i>mem64, xmm</i>	0F 13/ <i>r</i>	Moves two packed single-precision floating-point values from an XMM register to a 64-bit memory location.



Related Instructions

MOVAPS, MOVHLPS, MOVHPS, MOVLHPS, MOVMSKPS, MOVSS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

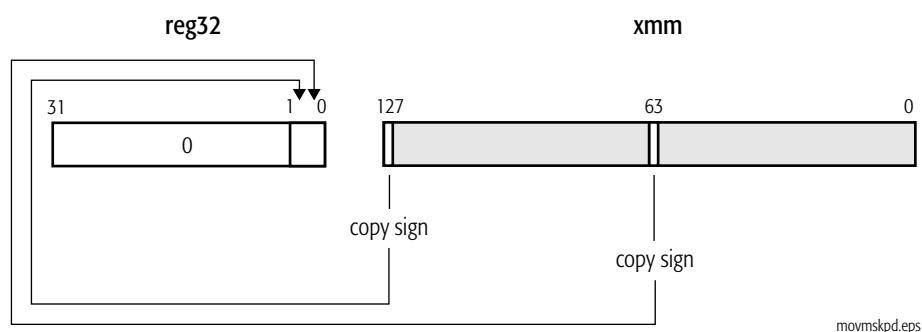
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of the control register (CR4) is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		One of the data operands falls outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

MOVMSKPD Extract Packed Double-Precision Floating-Point Sign Mask

The MOVMSKPD instruction moves the sign bits of two packed double-precision floating-point values in an XMM register to the two low-order bits of a general-purpose register, with zero-extension.

Mnemonic	Opcode	Description
MOVMSKPD <i>reg32, xmm</i>	66 0F 50 /r	Move sign bits in an XMM register to a 32-bit general-purpose register.
MOVMSKPD <i>reg64, xmm</i>	66 0F 50 /r	Move sign bits in an XMM register to a 64-bit general-purpose register.



movmskpd.eps

Related Instructions

MOVMSKPS, PMOVMSKB

rFLAGS Affected

None

MXCSR Flags Affected

None

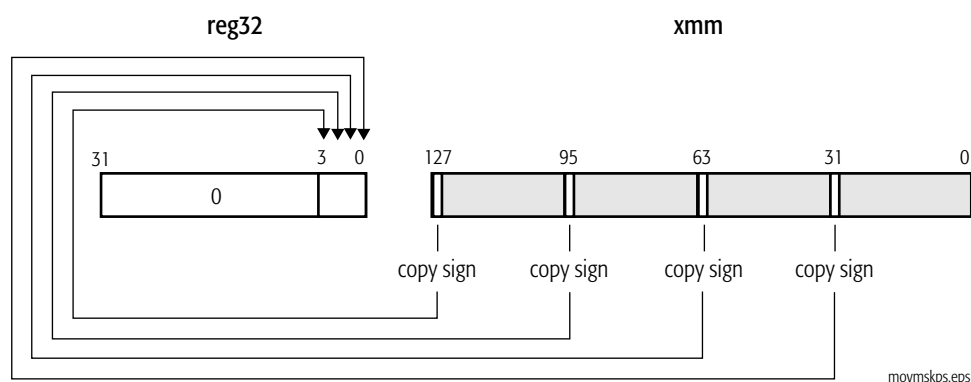
Exceptions

Exception (vector)	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.

MOVMSKPS Extract Packed Single-Precision Floating-Point Sign Mask

The MOVMSKPD instruction moves the sign bits of four packed single-precision floating-point values in an XMM register to the four low-order bits of a general-purpose register, with zero-extension.

Mnemonic	Opcode	Description
MOVMSKPS <i>reg32, xmm</i>	0F 50/ <i>r</i>	Move sign bits in an XMM register to a 32-bit general-purpose register.
MOVMSKPS <i>reg64, xmm</i>	0F 50/ <i>r</i>	Move sign bits in an XMM register to a 64-bit general-purpose register.



Related Instructions

MOVMSKPD, PMOVMSKB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

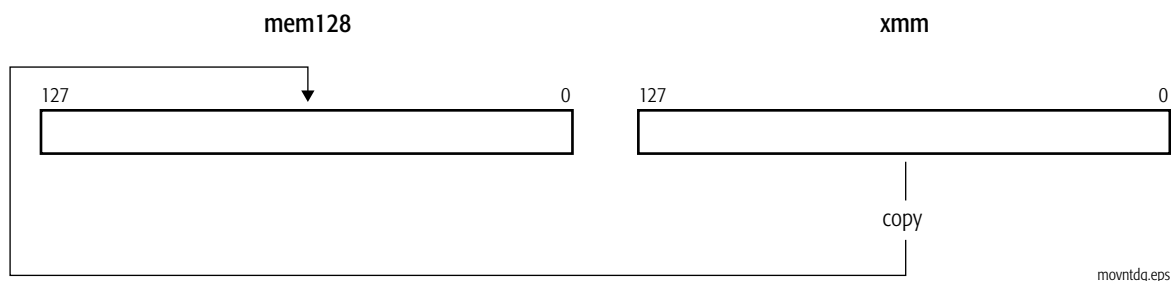
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.

MOVNTDQ**Move Non-Temporal Double Quadword**

The MOVNTDQ instruction stores a 128-bit (double quadword) XMM register value into a 128-bit memory location. This instruction indicates to the processor that the data is non-temporal, and is unlikely to be used again soon. The processor treats the store as a write-combining memory write, which minimizes cache pollution. The exact method by which cache pollution is minimized depends on the hardware implementation of the instruction. For further information, see “Memory Optimization” in Volume 1.

MOVNTDQ is weakly-ordered with respect to other instructions that operate on memory. Software should use an SFENCE instruction to force strong memory ordering of MOVNTDQ with respect to other stores.

Mnemonic	Opcode	Description
MOVNTDQ <i>mem128, xmm</i>	66 0F E7 /r	Stores a 128-bit XMM register value into a 128-bit memory location, minimizing cache pollution.

**Related Instructions**

MOVNTI, MOVNTPD, MOVNTPS, MOVNTQ

rFLAGS Affected

None

MXCSR Flags Affected

None

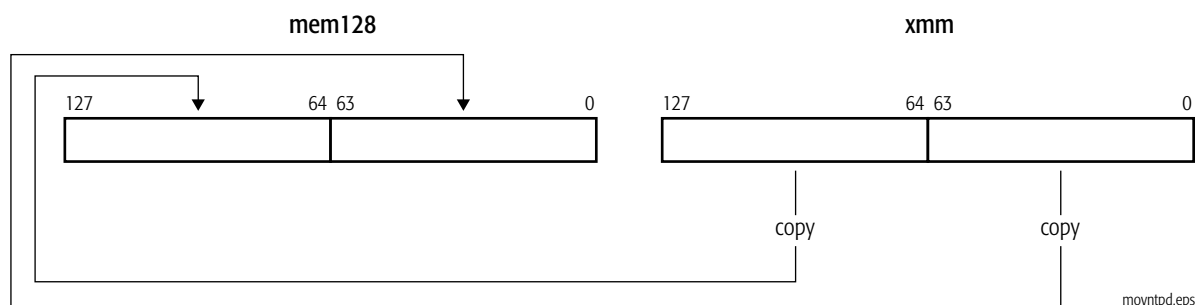
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (CR0.EM) is set to 1. The operating-system FXSAVE/FXRSTOR support bit (CR4.OSFXSR) is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (CR0.TS) is set to 1.
Stack, #SS			X	The operand's effective address points to an illegal memory location in the SS segment.
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, segment overrun, #GP	X	X		The effective address of a data operand is outside the address range 00000h to 0FFFFh.
General protection, #GP			X	The operand's effective address points to an illegal memory location in the CS, DS, ES, FS, or GS segment.
Page fault, #PF		X	X	A page fault resulted from executing the instruction.

MOVNTPD Move Non-Temporal Packed Double-Precision Floating-Point

The MOVNTPD instruction stores two double-precision floating-point XMM register values into a 128-bit memory location. This instruction indicates to the processor that the data is non-temporal, and is unlikely to be used again soon. The processor treats the store as a write-combining memory write, which minimizes cache pollution. The exact method by which cache pollution is minimized depends on the hardware implementation of the instruction. For further information, see “Memory Optimization” in Volume 1.

Mnemonic	Opcode	Description
MOVNTPD <i>mem128, xmm</i>	66 0F 2B /r	Stores two packed double-precision floating-point XMM register values into a 128-bit memory location, minimizing cache pollution.



MOVNTPD is weakly-ordered with respect to other instructions that operate on memory. Software should use an SFENCE instruction to force strong memory ordering of MOVNTPD with respect to other stores.

Related Instructions

MOVNTDQ, MOVNTI, MOVNTPS, MOVNTQ

rFLAGS Affected

None

MXCSR Flags Affected

None

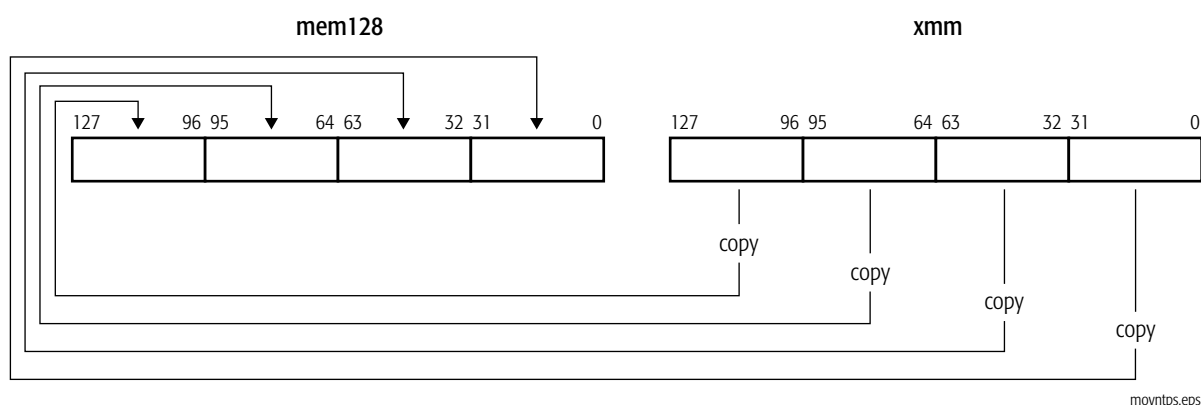
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (CR0.EM) is set to 1. The operating-system FXSAVE/FXRSTOR support bit (CR4.OSFXSR) is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (CR0.TS) is set to 1.
Stack, #SS			X	The operand's effective address points to an illegal memory location in the SS segment.
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, segment overrun, #GP	X	X		The effective address of a data operand is outside the address range 00000h to 0FFFFh.
General protection, #GP			X	The operand's effective address points to an illegal memory location in the CS, DS, ES, FS, or GS segment.
Page fault, #PF		X	X	A page fault resulted from executing the instruction.

MOVNTPS**Move Non-Temporal Packed
Single-Precision Floating-Point**

The MOVNTPS instruction stores four single-precision floating-point XMM register values into a 128-bit memory location. This instruction indicates to the processor that the data is non-temporal, and is unlikely to be used again soon. The processor treats the store as a write-combining memory write, which minimizes cache pollution. The exact method by which cache pollution is minimized depends on the hardware implementation of the instruction. For further information, see “Memory Optimization” in Volume 1.

Mnemonic	Opcode	Description
MOVNTPS <i>mem128, xmm</i>	0F 2B /r	Stores four packed single-precision floating-point XMM register values into a 128-bit memory location, minimizing cache pollution.



MOVNTPD is weakly-ordered with respect to other instructions that operate on memory. Software should use an SFENCE instruction to force strong memory ordering of MOVNTPD with respect to other stores.

Related Instructions

MOVNTDQ, MOVNTI, MOVNTPD, MOVNTQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

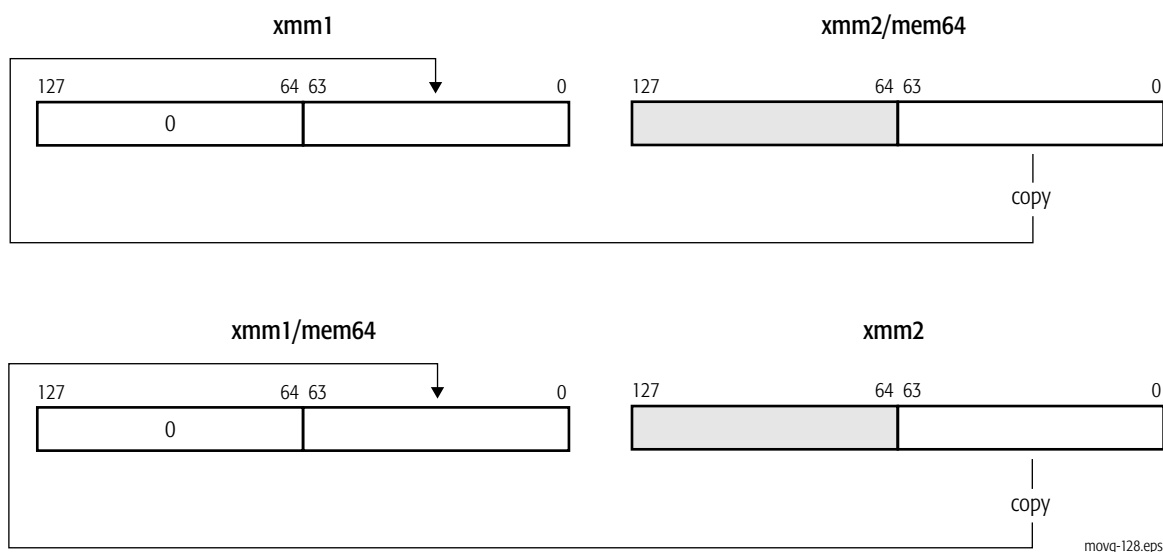
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (CR0.EM) is set to 1. The operating-system FXSAVE/FXRSTOR support bit (CR4.OSFXSR) is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (CR0.TS) is set to 1.
Stack, #SS			X	The operand's effective address points to an illegal memory location in the SS segment.
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, segment overrun, #GP	X	X		The effective address of a data operand is outside the address range 00000h to 0FFFFh.
General protection, #GP			X	The operand's effective address points to an illegal memory location in the CS, DS, ES, FS, or GS segment.
Page fault, #PF		X	X	A page fault resulted from executing the instruction.

MOVQ Move Quadword

The MOVQ instruction moves a 64-bit value in one of the following ways:

- from the low-order 64 bits of an XMM register or a 64-bit memory location to the low-order 64 bits of another XMM register, with zero-extension to 128 bits
- from the low-order 64 bits of an XMM register to the low-order 64 bits of another XMM register, with zero-extension to 128 bits or to a 64-bit memory location

Mnemonic	Opcode	Description
MOVQ <i>xmm1</i> , <i>xmm2/mem64</i>	F3 0F 7E	Moves 64-bit value from an XMM register or memory location to an XMM register.
MOVQ <i>xmm1/mem64</i> , <i>xmm2</i>	66 0F D6	Moves 64-bit value from an XMM register to an XMM register or memory location.



Related Instructions

MOVD, MOVDQA, MOVDQU, MOVDQ2Q, MOVQ2DQ

rFLAGS Affected

None

MXCSR Flags Affected

None

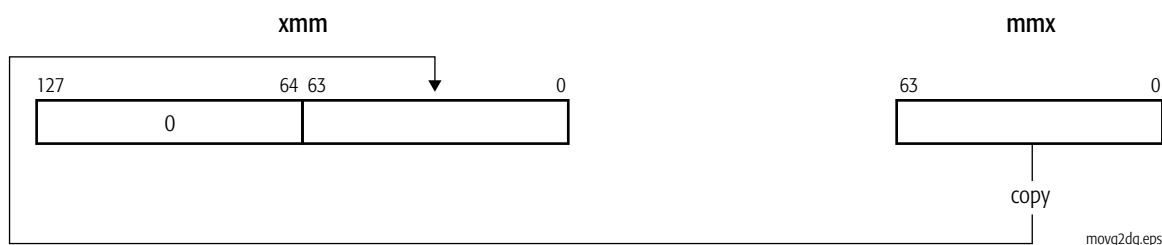
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

MOVQ2DQ**Move Quadword to Quadword**

The MOVQ2DQ instruction moves a 64-bit value from an MMX register to the low-order 64 bits of an XMM register, with zero-extension to 128 bits.

Mnemonic	Opcode	Description
MOVQ2DQ <i>xmm, mmx</i>	F3 0F D6	Moves 64-bit value from an MMX™ register to an XMM register.



Execution of this instruction causes all fields in the x87 tag word to be set according to their corresponding data, the top-of-stack-pointer bit (TOP) in the x87 status word to be cleared to 0, and any pending x87 exceptions are handled before this instruction is executed. For details, see “Actions Taken on Executing 64-Bit Media Instructions” in Volume 1.

Related Instructions

MOVD, MOVDQA, MOVDQU, MOVDQ2Q, MOVQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
x87 floating-point exception pending, #MF	X	X	X	An exception is pending due to an x87 floating-point instruction.

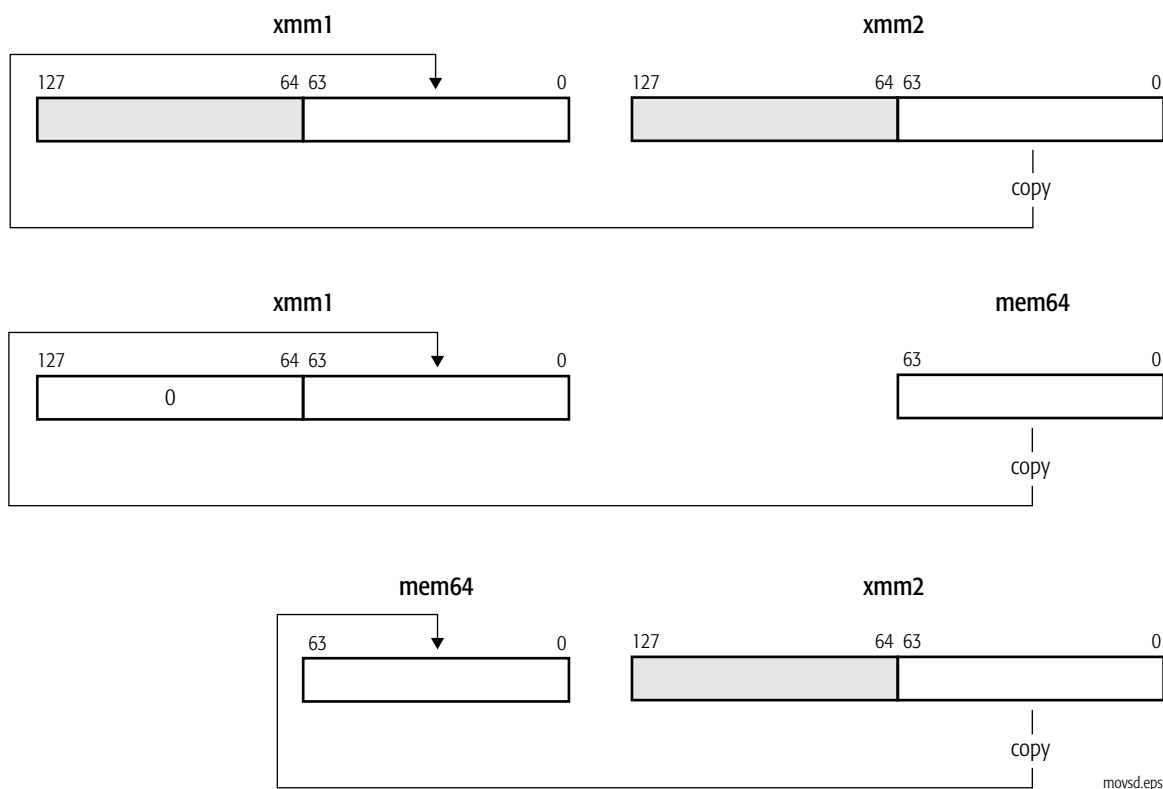
Move Scalar Double-Precision Floating-Point

The MOVSD instruction moves a scalar double-precision floating-point value:

- from the low-order 64 bits of an XMM register or a 64-bit memory location to the low-order 64 bits of another XMM register, or
- from the low-order 64 bits of an XMM register to the low-order 64 bits of another XMM register or a 64-bit memory location.

If the source operand is an XMM register, the high-order 64 bits of the destination XMM register are not modified. If the source operand is a memory location, the high-order 64 bits of the destination XMM register are cleared to all 0s.

Mnemonic	Opcode	Description
MOVSD <i>xmm1, xmm2/mem64</i>	F2 0F 10 /r	Moves double-precision floating-point value from an XMM register or 64-bit memory location to an XMM register.
MOVSD <i>xmm1/mem64, xmm2</i>	F2 0F 11 /r	Moves double-precision floating-point value from an XMM register to an XMM register or 64-bit memory location.



This MOVSD instruction should not be confused with the same-mnemonic MOVSD (move string doubleword) instruction in the general-purpose instruction set. Assemblers can distinguish the instructions by the number and type of operands.

Related Instructions

MOVAPD, MOVHPD, MOVLPD, MOVMSKPD, MOVUPD

rFLAGS Affected

None.

MXCSR Flags Affected

None.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

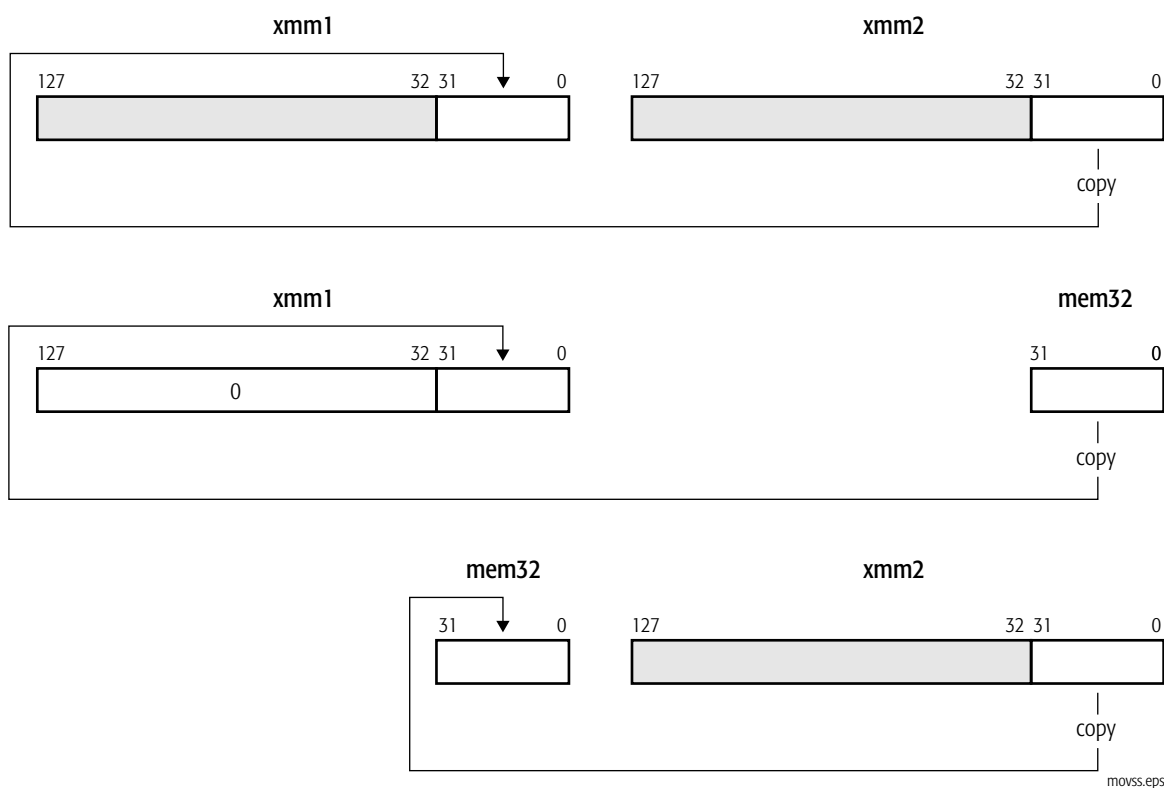
MOVSS**Move Scalar Single-Precision Floating-Point**

The MOVSS instruction moves a scalar single-precision floating-point value:

- from the low-order 32 bits of an XMM register or a 32-bit memory location to the low-order 32 bits of another XMM register, or
- from a 32-bit memory location to the low-order 32 bits of an XMM register, with zero-extension to 128 bits.

If the source operand is an XMM register, the high-order 96 bits of the destination XMM register are not modified. If the source operand is a memory location, the high-order 96 bits of the destination XMM register are cleared to all 0s.

Mnemonic	Opcode	Description
MOVSS <i>xmm1, xmm2/mem32</i>	F3 0F 10 / <i>r</i>	Moves single-precision floating-point value from an XMM register or 32-bit memory location to an XMM register.
MOVSS <i>xmm1/mem32, xmm2</i>	F3 0F 11 / <i>r</i>	Moves single-precision floating-point value from an XMM register to an XMM register or 32-bit memory location.



Related Instructions

MOVAPS, MOVHLPS, MOVHPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

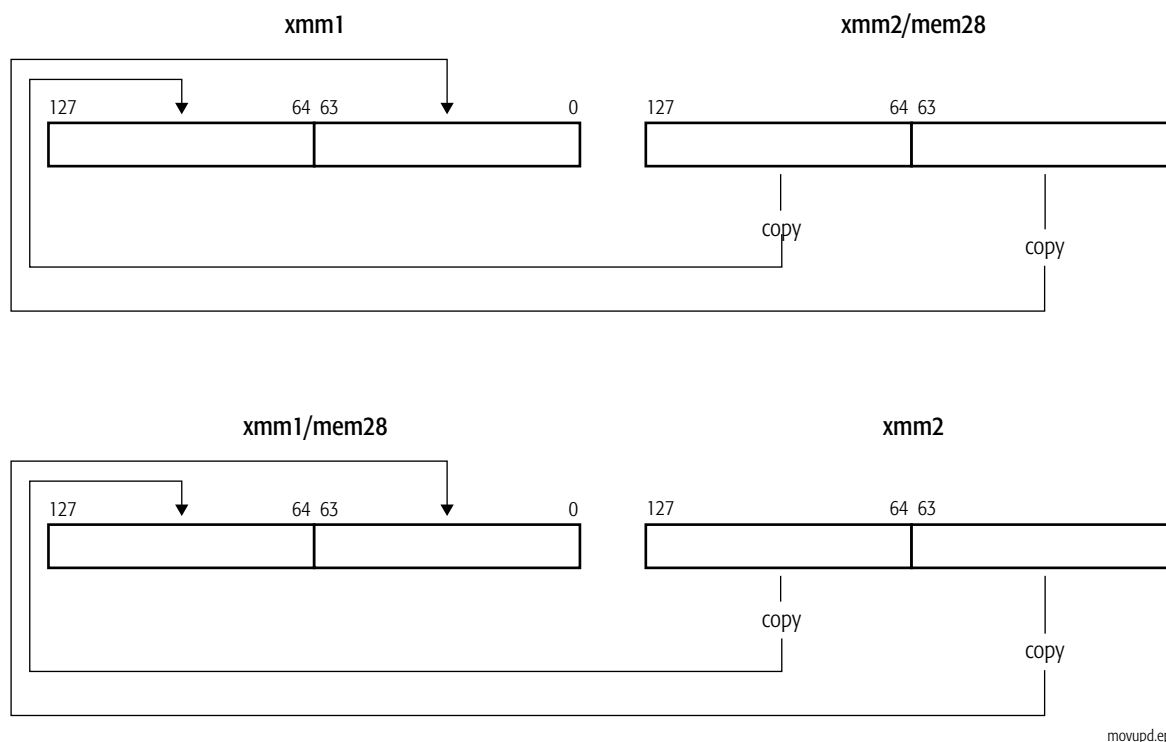
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

MOVUPD Move Unaligned Packed Double-Precision Floating-Point

The MOVUPD instruction moves two packed double-precision floating-point values:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
MOVUPD <i>xmm1, xmm2/mem128</i>	66 0F 10 /r	Moves two unaligned packed double-precision floating-point values from an XMM register or 128-bit memory location to an XMM register.
MOVUPD <i>xmm1/mem128, xmm2</i>	66 0F 11 /r	Moves two unaligned packed double-precision floating-point values from an XMM register to an XMM register or 128-bit memory location.



Memory operands that are not aligned on a 16-byte boundary do not cause a general-protection exception.

Related Instructions

MOVAPD, MOVHPD, MOVLPD, MOVMSKPD, MOVSD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

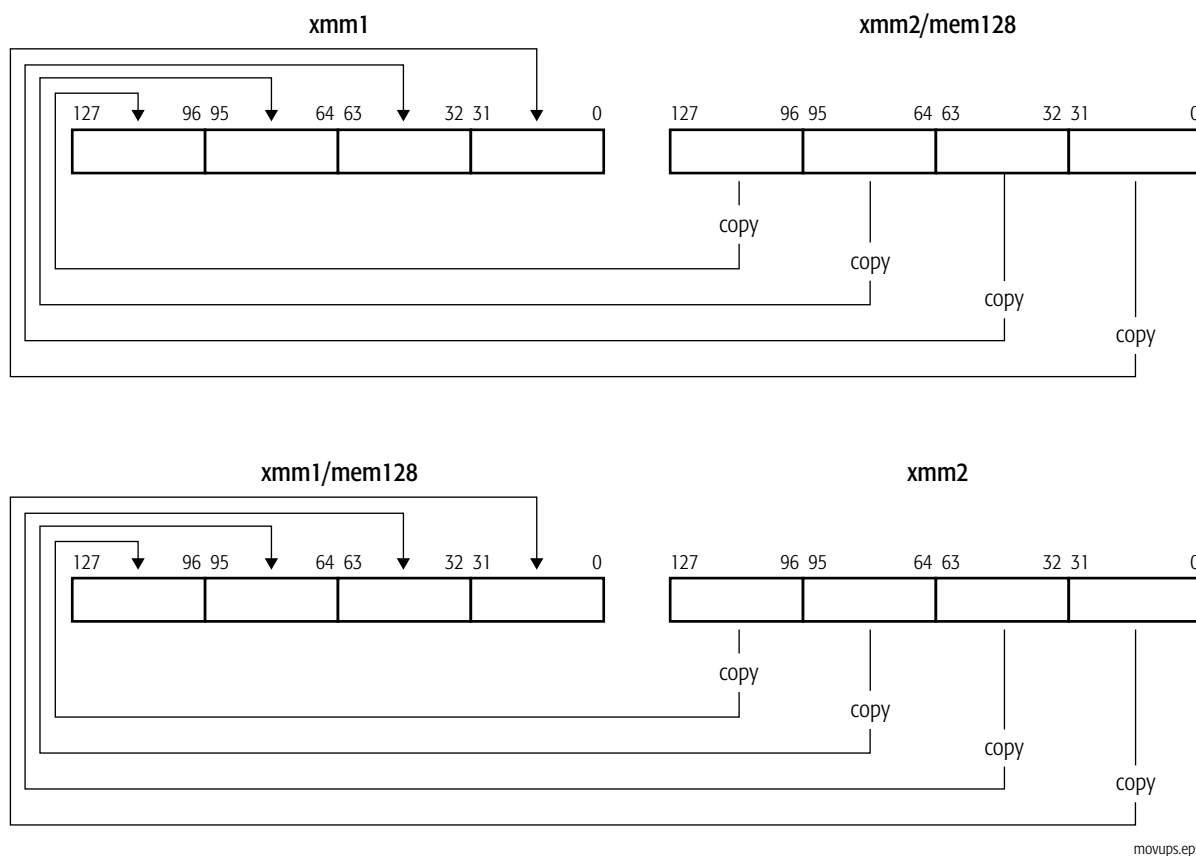
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

MOVUPS Move Unaligned Packed Single-Precision Floating-Point

The MOVUPS instruction moves four packed single-precision floating-point values:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
MOVUPS <i>xmm1</i> , <i>xmm2/mem128</i>	0F 10 /r	Moves four unaligned packed single-precision floating-point values from an XMM register or 128-bit memory location to an XMM register.
MOVUPS <i>xmm1/mem128</i> , <i>xmm2</i>	0F 11 /r	Moves four unaligned packed single-precision floating-point values from an XMM register to an XMM register or 128-bit memory location.



Memory operands that are not aligned on a 16-byte boundary do not cause a general-protection exception.

Related Instructions

MOVAPS, MOVHLPS, MOVHPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVSS

rFLAGS Affected

None

MXCSR Flags Affected

None

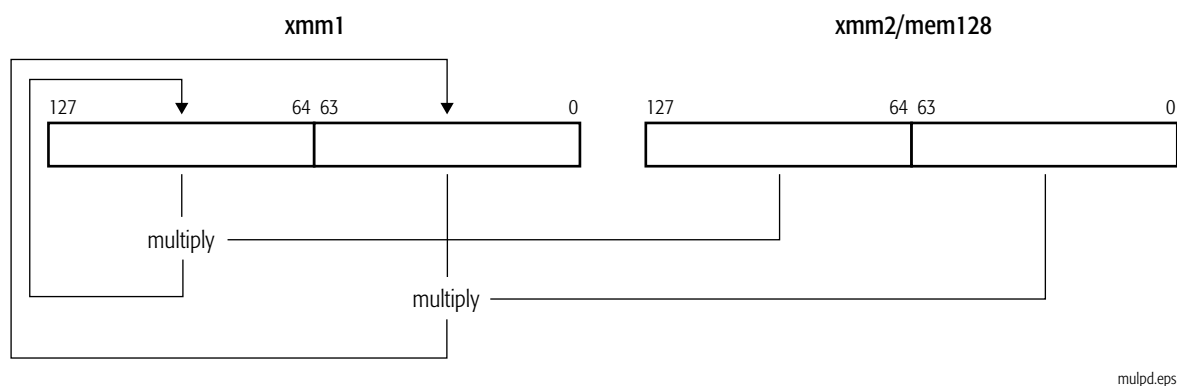
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

MULPD Multiply Packed Double-Precision Floating-Point

The MULPD instruction multiplies each of the two packed double-precision floating-point values in the first source operand by the corresponding packed double-precision floating-point value in the second source operand and writes the result of each multiplication operation in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
MULPD <i>xmm1, xmm2/mem128</i>	66 0F 59 /r	Multiplies packed double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the results in the destination XMM register.



Related Instructions

MULPS, MULSD, MULSS, PFMUL

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M		M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

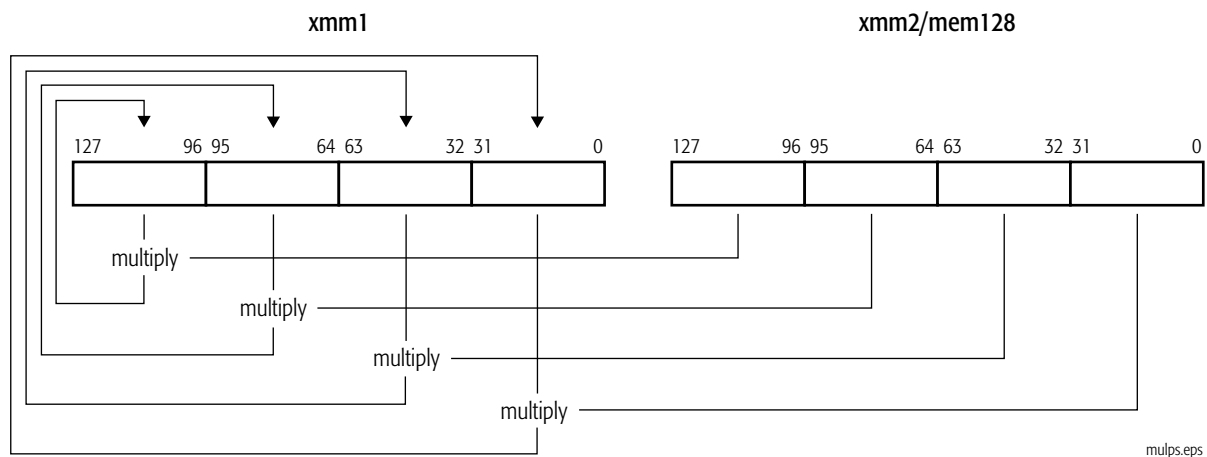
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

MULPS**Multiply Packed Single-Precision Floating-Point**

The MULPS instruction multiplies each of the four packed single-precision floating-point values in first source operand by the corresponding packed single-precision floating-point value in the second source operand and writes the result of each multiplication operation in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
MULPS <i>xmm1, xmm2/mem128</i>	0F 59/r	Multiplies packed single-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the results in the destination XMM register.

**Related Instructions**

MULPD, MULSD, MULSS, PFMUL

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M		M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.

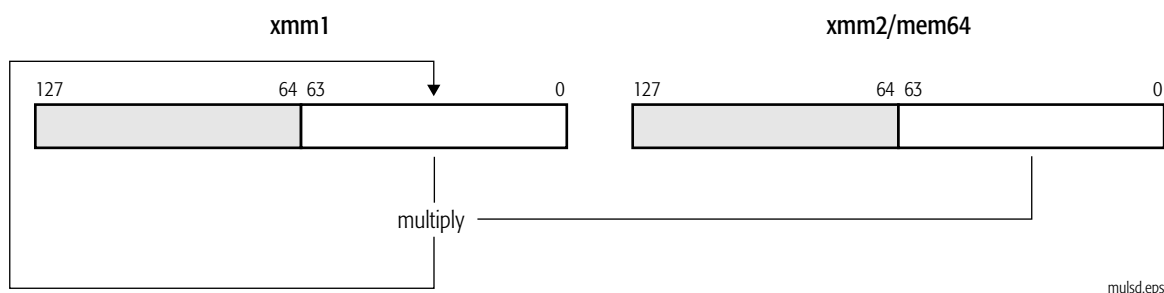
Exception	Real	Virtual 8086	Protected	Cause of Exception
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

MULSD

Multiply Scalar Double-Precision Floating-Point

The MULSD instruction multiplies the double-precision floating-point value in the low-order quadword of first source operand by the double-precision floating-point value in the low-order quadword of the second source operand and writes the result in the low-order quadword of the destination (first source). The high-order quadword of the destination is not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 64-bit memory location.

Mnemonic	Opcode	Description
MULSD <i>xmm1, xmm2/mem64</i>	F2 0F 59 /r	Multiplies low-order double-precision floating-point values in an XMM register and another XMM register or 64-bit memory location and writes the result in the low-order quadword of the destination XMM register.



Related Instructions

MULPD, MULPS, MULSS, PFMUL

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M		M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

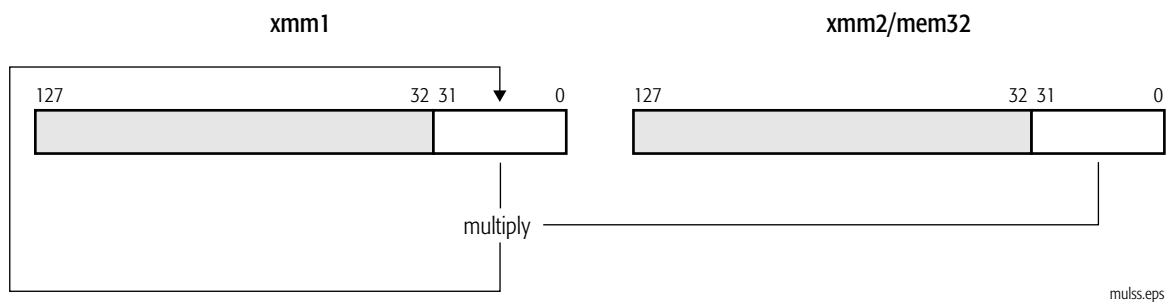
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

MULSS Multiply Scalar Single-Precision Floating-Point

The MULSS instruction multiplies the single-precision floating-point value in the low-order doubleword of first source operand by the single-precision floating-point value in the low-order doubleword of the second source operand and writes the result in the low-order doubleword of the destination (first source). The three high-order doublewords of the destination are not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 32-bit memory location.

Mnemonic	Opcode	Description
MULSS <i>xmm1, xmm2/mem32</i>	F3 0F 59/r	Multiplies low-order single-precision floating-point values in an XMM register and another XMM register or 32-bit memory location and writes the result in the low-order doubleword of the destination XMM register.



Related Instructions

MULPD, MULPS, MULSD, PFMUL

rFLAGS Affected

None

MXCSR Flags Affected.

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M		M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

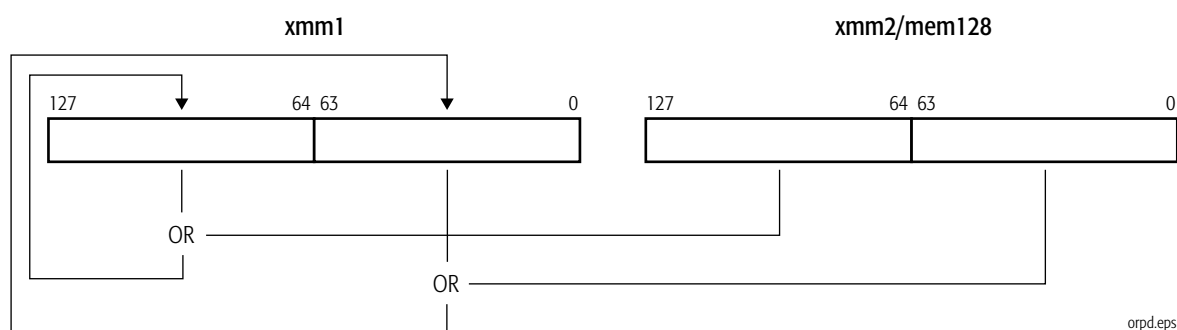
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

ORPD**Logical Bitwise OR
Packed Double-Precision Floating-Point**

The ORPD instruction performs a bitwise logical OR of the two packed double-precision floating-point values in the first source operand and the corresponding two packed double-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
ORPD <i>xmm1, xmm2/mem128</i>	66 0F 56/r	Performs bitwise logical OR of two packed double-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.

**Related Instructions**

ANDNPD, ANDNPS, ANDPD, ANDPS, ORPS, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

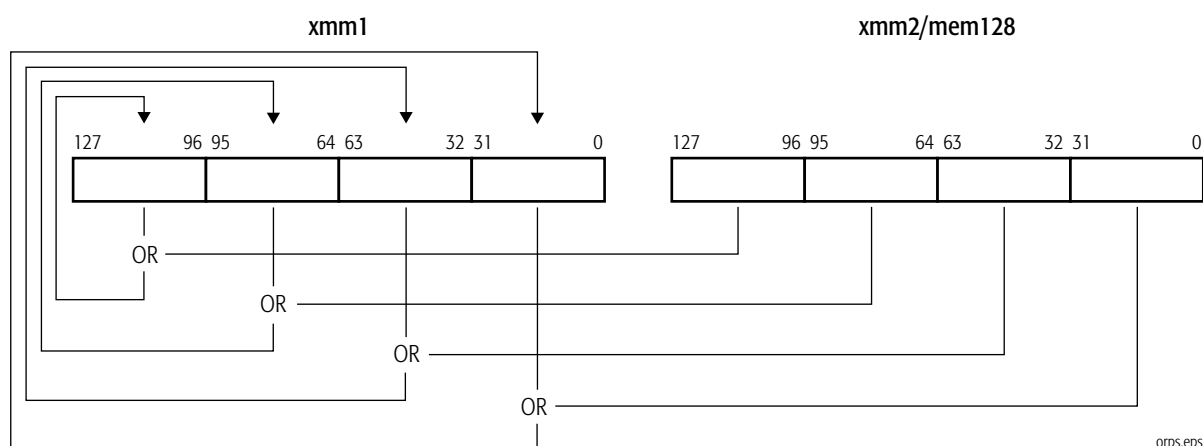
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

ORPS**Logical Bitwise OR
Packed Single-Precision Floating-Point**

The ORPS instruction performs a bitwise logical OR of the four packed single-precision floating-point values in the first source operand and the corresponding four packed single-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
ORPS <i>xmm1, xmm2/mem128</i>	OF 56 /r	Performs bitwise logical OR of four packed single-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.

**Related Instructions**

ANDNPD, ANDNPS, ANDPD, ANDPS, ORPD, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

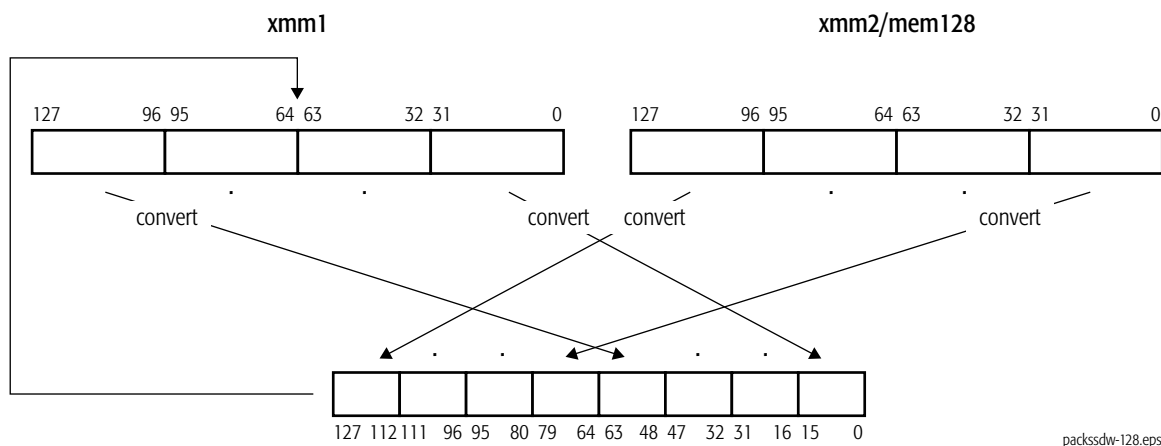
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PACKSSDW**Pack with Saturation Signed Doubleword to Word**

The PACKSSDW instruction converts each 32-bit signed integer in the first and second source operands to a 16-bit signed integer and packs the converted values into words in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Converted values from the first source operand are packed into the low-order words of the destination, and the converted values from the second source operand are packed into the high-order words of the destination.

Mnemonic	Opcode	Description
PACKSSDW <i>xmm1, xmm2/mem128</i>	66 0F 6B /r	Packs 32-bit signed integers in an XMM register and another XMM register or 128-bit memory location into 16-bit signed integers in an XMM register.



packssdw-128.eps

For each packed value in the destination, if the value is larger than the largest signed 16-bit integer, it is saturated to 7FFFh, and if the value is smaller than the smallest signed 16-bit integer, it is saturated to 8000h.

Related Instructions

PACKSSWB, PACKUSWB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

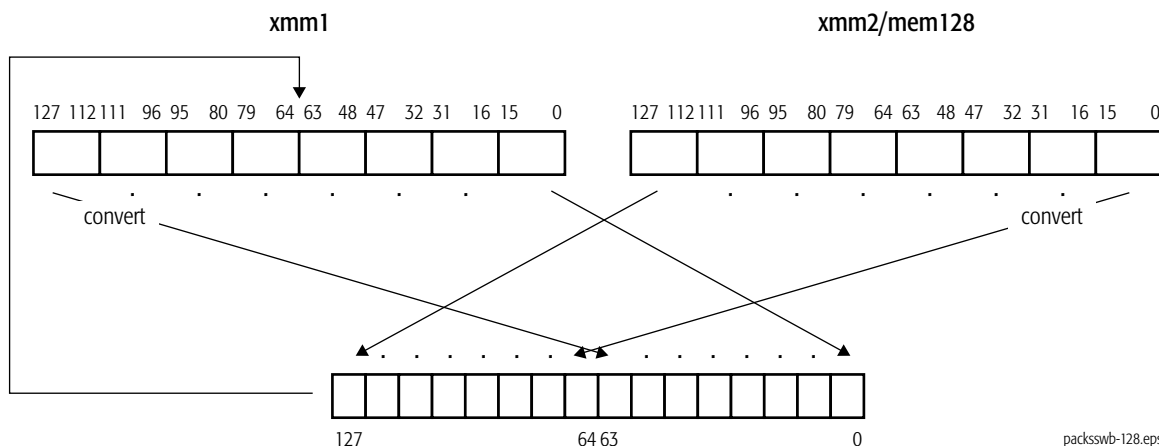
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PACKSSWB**Pack with Saturation Signed Word to Byte**

The PACKSSWB instruction converts each 16-bit signed integer in the first and second source operands to an 8-bit signed integer and packs the converted values into bytes in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Converted values from the first source operand are packed into the low-order bytes of the destination, and the converted values from the second source operand are packed into the high-order bytes of the destination.

Mnemonic	Opcode	Description
PACKSSWB <i>xmm1, xmm2/mem128</i>	66 0F 63 /r	Packs 16-bit signed integers in an XMM register and another XMM register or 128-bit memory location into 8-bit signed integers in an XMM register.



For each packed value in the destination, if the value is larger than the largest signed 8-bit integer, it is saturated to 7Fh, and if the value is smaller than the smallest signed 8-bit integer, it is saturated to 80h.

Related Instructions

PACKSSDW, PACKUSWB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

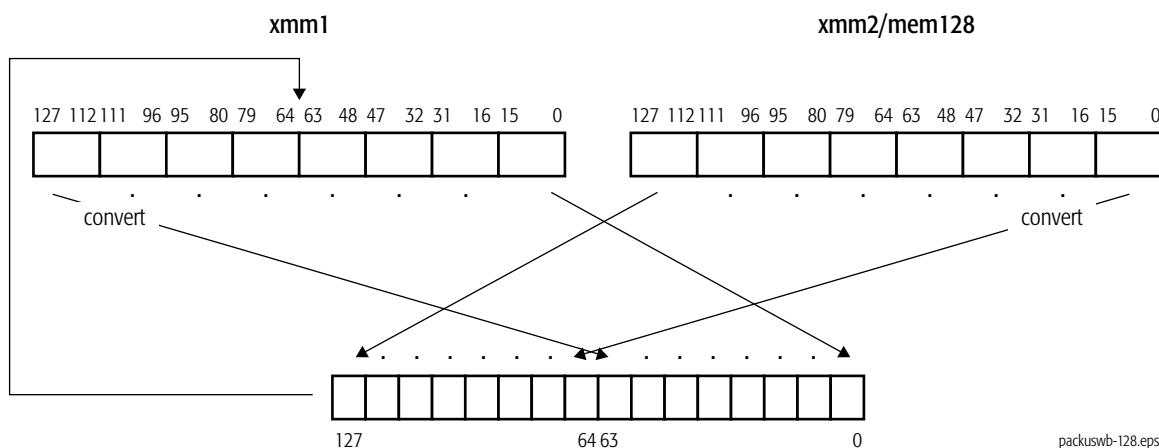
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PACKUSWB Pack with Saturation Signed Word to Unsigned Byte

The PACKUSWB instruction converts each 16-bit signed integer in the first and second source operands to an 8-bit unsigned integer and packs the converted values into bytes in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Converted values from the first source operand are packed into the low-order bytes of the destination, and the converted values from the second source operand are packed into the high-order bytes of the destination.

Mnemonic	Opcode	Description
PACKUSWB <i>xmm1, xmm2/mem128</i>	66 0F 67 /r	Packs 16-bit signed integers in an XMM register and another XMM register or 128-bit memory location into 8-bit unsigned integers in an XMM register.



For each packed value in the destination, if the value is larger than the largest unsigned 8-bit integer, it is saturated to FFh, and if the value is smaller than the smallest unsigned 8-bit integer, it is saturated to 00h.

Related Instructions

PACKSSDW, PACKSSWB

rFLAGS Affected

None

MXCSR Flags Affected

None

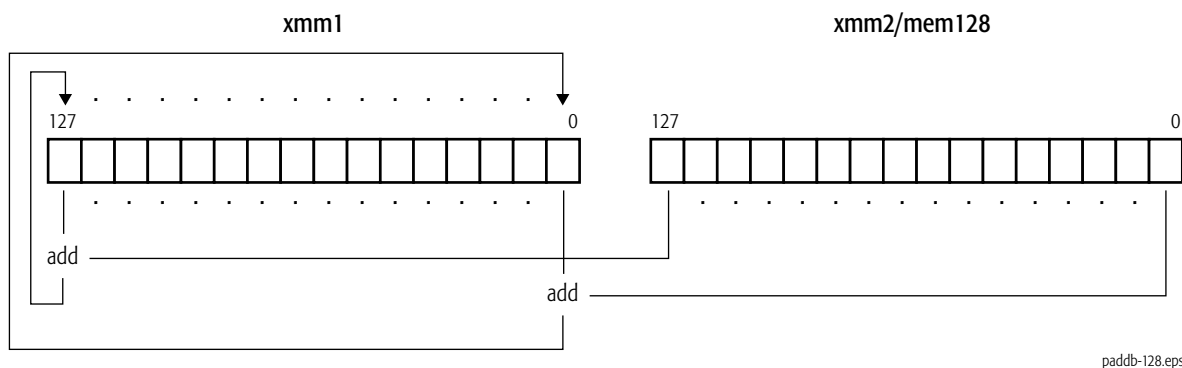
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDB Packed Add Bytes

The PADDB instruction adds each packed 8-bit integer value in the first source operand to the corresponding packed 8-bit integer in the second source operand and writes the integer result of each addition in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PADDB <i>xmm1, xmm2/mem128</i>	66 0F FC /r	Adds packed byte integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 8 bits of each result are written in the destination.

Related Instructions

PADDD, PADDQ, PADDSB, PADDSW, PADDUSB, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

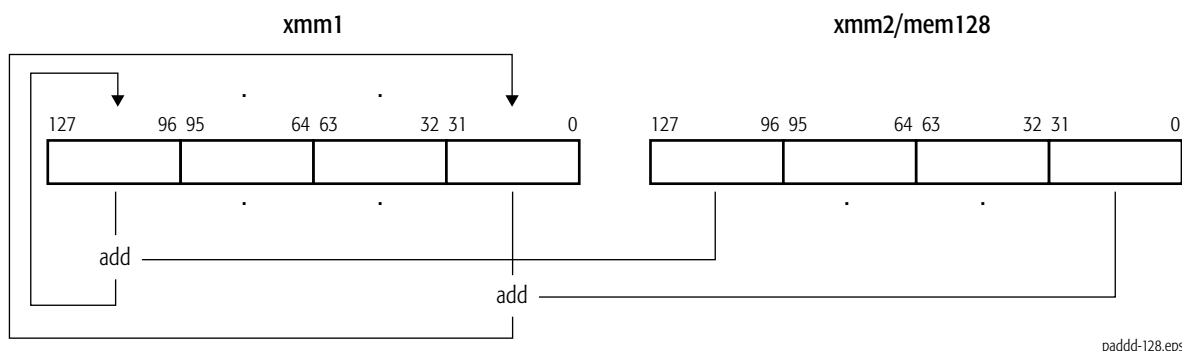
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADD**Packed Add Doublewords**

The PADD instruction adds each packed 32-bit integer value in the first source operand to the corresponding packed 32-bit integer in the second source operand and writes the integer result of each addition in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PADD <i>xmm1, xmm2/mem128</i>	66 0F FE /r	Adds packed 32-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 32 bits of each result are written in the destination.

Related Instructions

PADB, PADDQ, PADDSB, PADDSW, PADDUSB, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

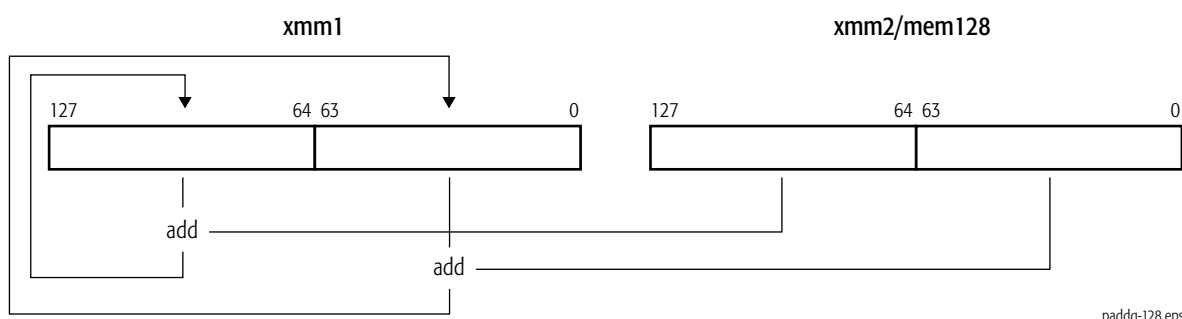
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDQ**Packed Add Quadwords**

The PADDQ instruction adds each packed 64-bit integer value in the first source operand to the corresponding packed 64-bit integer in the second source operand and writes the integer result of each addition in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PADDQ <i>xmm1, xmm2/mem128</i>	66 0F D4 /r	Adds packed 64-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 64 bits of each result are written in the destination.

Related Instructions

PADB, PADDD, PADDSB, PADDSW, PADDUSB, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

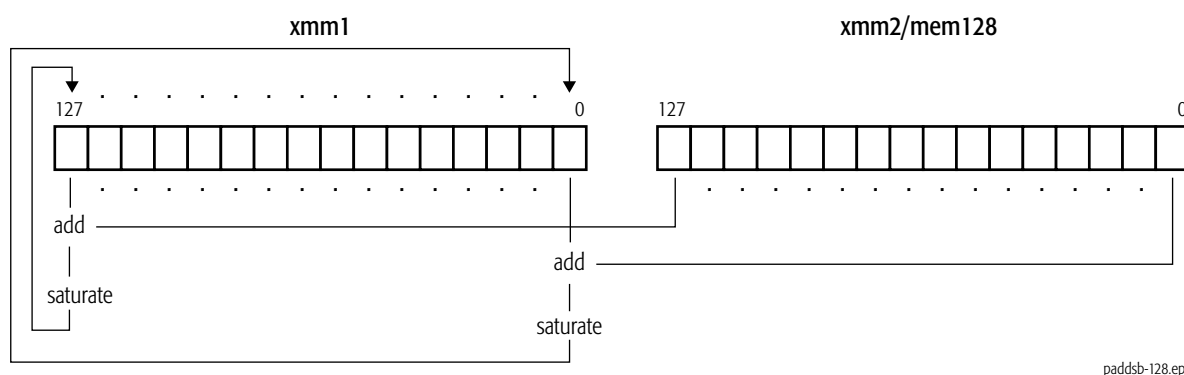
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDSB Packed Add Signed with Saturation Bytes

The PADDSB instruction adds each packed 8-bit signed integer value in the first source operand to the corresponding packed 8-bit signed integer in the second source operand and writes the signed integer result of each addition in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PADDSB <i>xmm1, xmm2/mem128</i>	66 0F EC /r	Adds packed byte signed integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



For each packed value in the destination, if the value is larger than the largest representable signed 8-bit integer, it is saturated to 7Fh, and if the value is smaller than the smallest signed 8-bit integer, it is saturated to 80h.

Related Instructions

PADDB, PADDD, PADDQ, PADDSW, PADDUSB, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

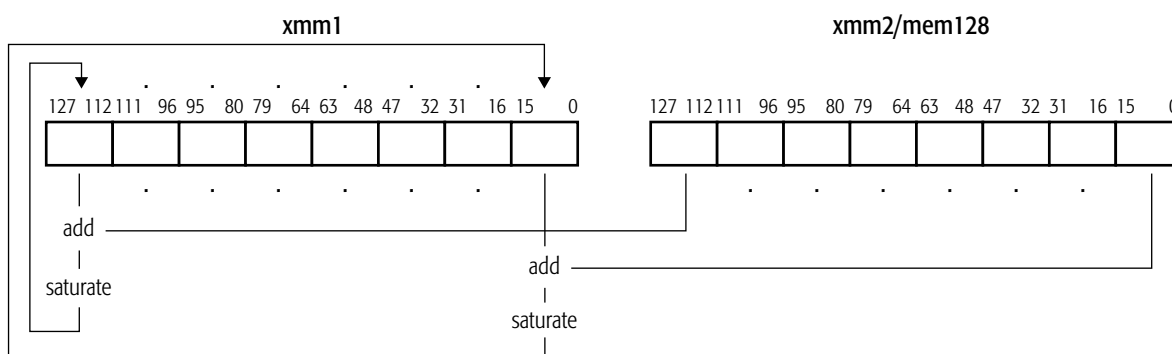
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDSW**Packed Add Signed with Saturation Words**

The PADDSW instruction adds each packed 16-bit signed integer value in the first source operand to the corresponding packed 16-bit signed integer in the second source operand and writes the signed integer result of each addition in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PADDSW <i>xmm1, xmm2/mem128</i>	66 0F ED /r	Adds packed 16-bit signed integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



paddsw-128.eps

For each packed value in the destination, if the value is larger than the largest representable signed 16-bit integer, it is saturated to 7FFFh, and if the value is smaller than the smallest signed 16-bit integer, it is saturated to 8000h.

Related Instructions

PADDB, PADDD, PADDQ, PADDSB, PADDUSB, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

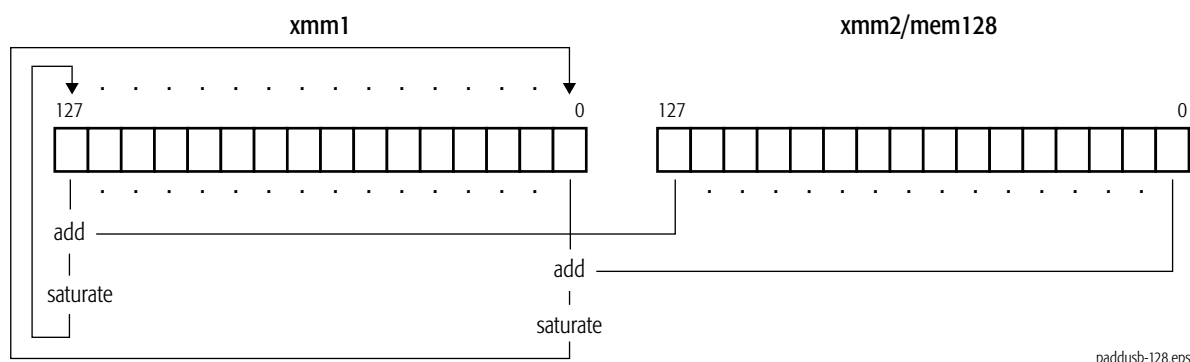
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDUSB**Packed Add Unsigned with Saturation Bytes**

The PADDUSB instruction adds each packed 8-bit unsigned integer value in the first source operand to the corresponding packed 8-bit unsigned integer in the second source operand and writes the unsigned integer result of each addition in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PADDUSB <i>xmm1, xmm2/mem128</i>	66 0F DC /r	Adds packed byte unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



For each packed value in the destination, if the value is larger than the largest unsigned 8-bit integer, it is saturated to FFh, and if the value is smaller than the smallest unsigned 8-bit integer, it is saturated to 00h.

Related Instructions

PADDB, PADDD, PADDQ, PADDSB, PADDSW, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

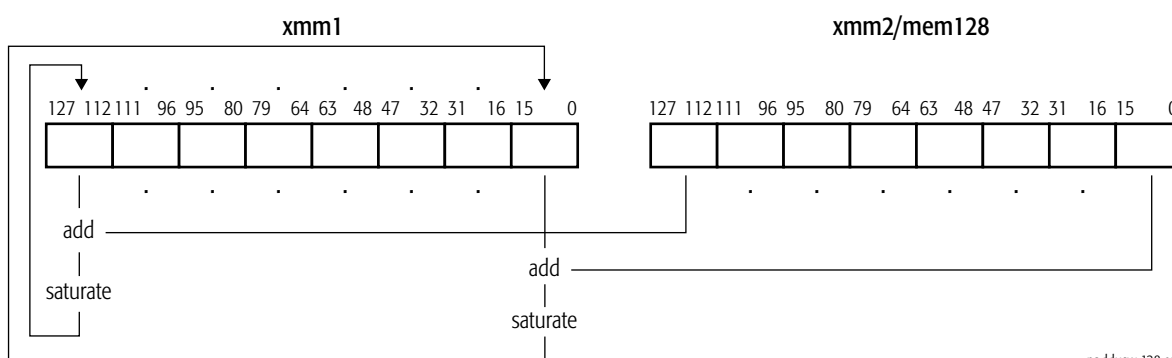
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDUSW**Packed Add Unsigned with Saturation Words**

The PADDUSW instruction adds each packed 16-bit unsigned integer value in the first source operand to the corresponding packed 16-bit unsigned integer in the second source operand and writes the unsigned integer result of each addition in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PADDUSW <i>xmm1</i> , <i>xmm2/mem128</i>	66 0F DD /r	Adds packed 16-bit unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes result in the destination XMM register.



paddusw-128.eps

For each packed value in the destination, if the value is larger than the largest unsigned 16-bit integer, it is saturated to FFFFh, and if the value is smaller than the smallest unsigned 16-bit integer, it is saturated to 0000h.

Related Instructions

PADDB, PADDD, PADDQ, PADDSB, PADDSW, PADDUSB, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

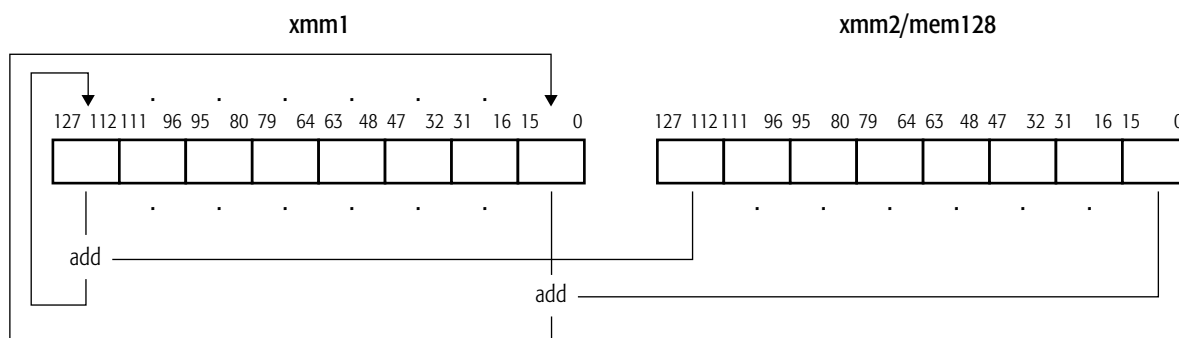
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDW**Packed Add Words**

The PADDW instruction adds each packed 16-bit integer value in the first source operand to the corresponding packed 16-bit integer in the second source operand and writes the integer result of each addition in the corresponding word of the destination (second source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PADDW <i>xmm1, xmm2/mem128</i>	66 0F FD /r	Adds packed 16-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



paddw-128.eps

This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 16 bits of the result are written in the destination.

Related Instructions

PADDB, PADDD, PADDQ, PADDSB, PADDSW, PADDUSB, PADDUSW

rFLAGS Affected

None

MXCSR Flags Affected

None

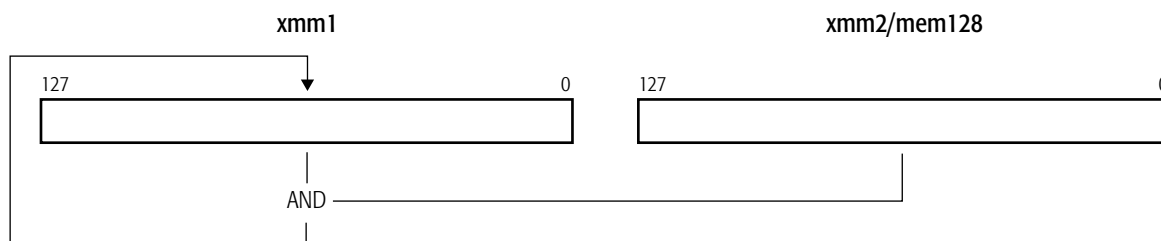
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PAND**Packed Logical Bitwise AND**

The PAND instruction performs a bitwise logical AND of the values in the first and second source operands and writes the result in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PAND <i>xmm1, xmm2/mem128</i>	66 0F DB /r	Performs bitwise logical AND of values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



pand-128.eps

Related Instructions

PANDN, POR, PXOR

rFLAGS Affected

None

MXCSR Flags Affected

None

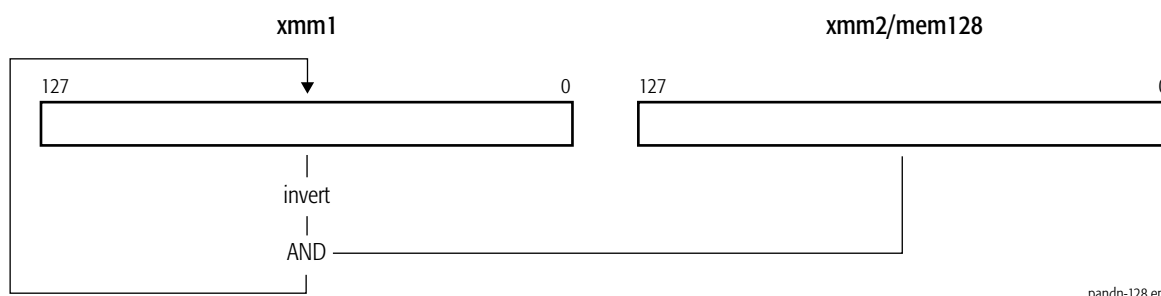
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PANDN**Packed Logical Bitwise AND NOT**

The PANDN instruction performs a bitwise logical AND of the value in the second source operand and the one's complement of the value in the first source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PANDN <i>xmm1, xmm2/mem128</i>	66 0F DF /r	Performs bitwise logical AND NOT of values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



pandn-128.eps

Related Instructions

PAND, POR, PXOR

rFLAGS Affected

None

MXCSR Flags Affected

None

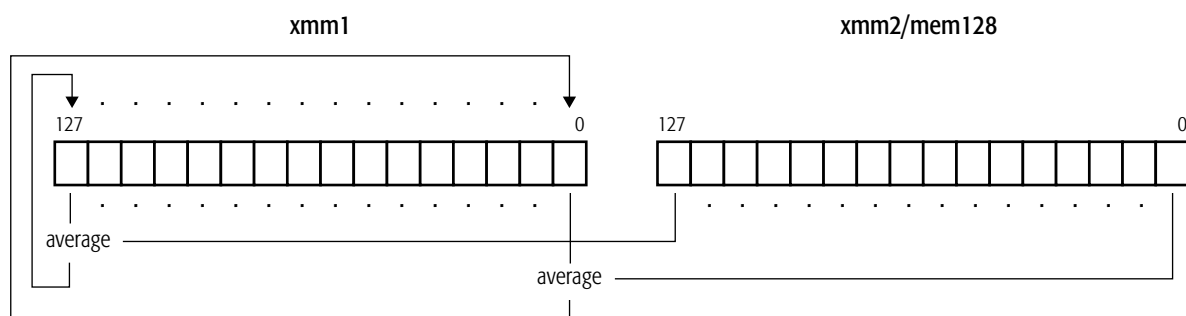
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PAVGB**Packed Average Unsigned Bytes**

The PAVGB instruction computes the rounded average of each packed unsigned 8-bit integer value in the first source operand and the corresponding packed 8-bit unsigned integer in the second source operand and writes each average in the corresponding byte of the destination (first source). The average is computed by adding each pair of operands, adding 1 to the 9-bit temporary sum, and then right-shifting the temporary sum by one bit position. The destination and source operands are an XMM register and another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PAVGB <i>xmm1, xmm2/mem128</i>	66 0F E0 /r	Averages packed 8-bit unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



pavgb-128.eps

Related Instructions

PAVGW

rFLAGS Affected

None

MXCSR Flags Affected

None

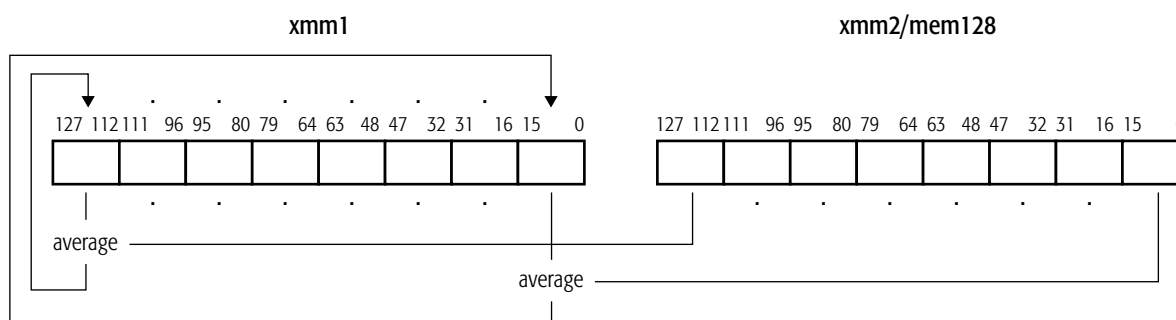
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PAVGW**Packed Average Unsigned Words**

The PAVGW instruction computes the rounded average of each packed unsigned 16-bit integer value in the first source operand and the corresponding packed 16-bit unsigned integer in the second source operand and writes each average in the corresponding word of the destination (first source). The average is computed by adding each pair of operands, adding 1 to the 17-bit temporary sum, and then right-shifting the temporary sum by one bit position. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PAVGW <i>xmm1, xmm2/mem128</i>	66 0F E3 /r	Averages packed 16-bit unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



pavgw-128.eps

Related Instructions

PAVGB

rFLAGS Affected

None

MXCSR Flags Affected

None

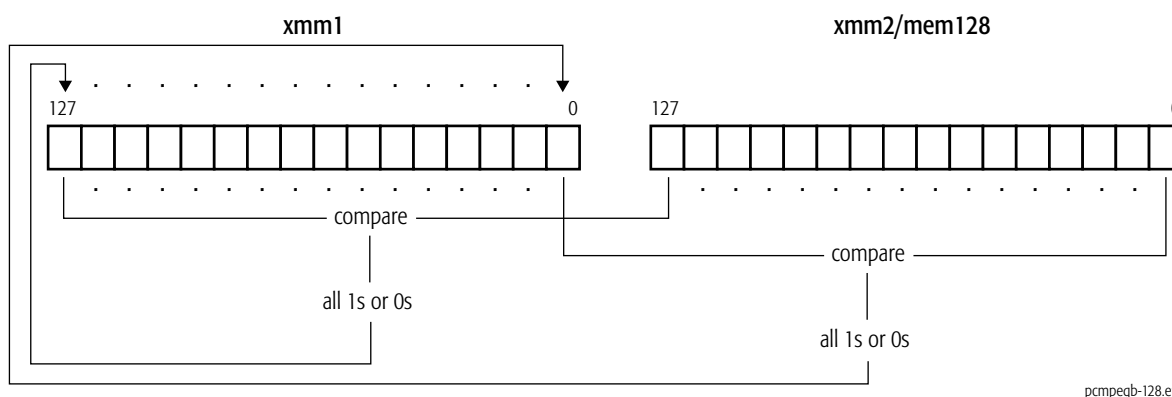
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPEQB**Packed Compare Equal Bytes**

The PCMPEQB instruction compares corresponding packed bytes in the first and second source operands and writes the result of each comparison in the corresponding byte of the destination (first source). For each pair of bytes, if the values are equal, the result is all 1s. If the values are not equal, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PCMPEQB <i>xmm1, xmm2/mem128</i>	66 0F 74 /r	Compares packed bytes in an XMM register and an XMM register or 128-bit memory location.



pcmpeqb-128.eps

Related Instructions

PCMPEQD, PCMPEQW, PCMPGTB, PCMPGTD, PCMPGTW

rFLAGS Affected

None

MXCSR Flags Affected

None

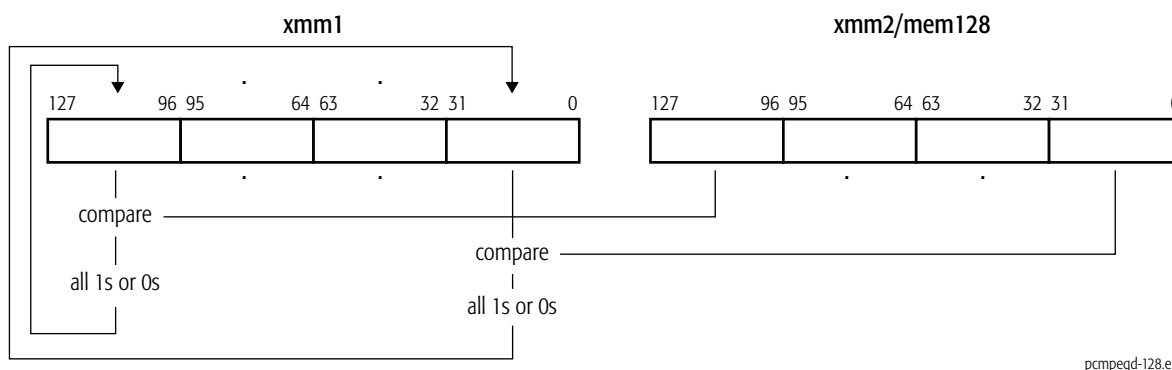
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPEQD**Packed Compare Equal Doublewords**

The PCMPEQD instruction compares corresponding packed 32-bit values in the first and second source operands and writes the result of each comparison in the corresponding 32 bits of the destination (first source). For each pair of doublewords, if the values are equal, the result is all 1s. If the values are not equal, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PCMPEQD <i>xmm1, xmm2/mem128</i>	66 0F 76 /r	Compares packed doublewords in an XMM register and an XMM register or 128-bit memory location.



pcmpeq-d-128.eps

Related Instructions

PCMPEQB, PCMPEQW, PCMPGTB, PCMPGTD, PCMPGTW

rFLAGS Affected

None

MXCSR Flags Affected

None

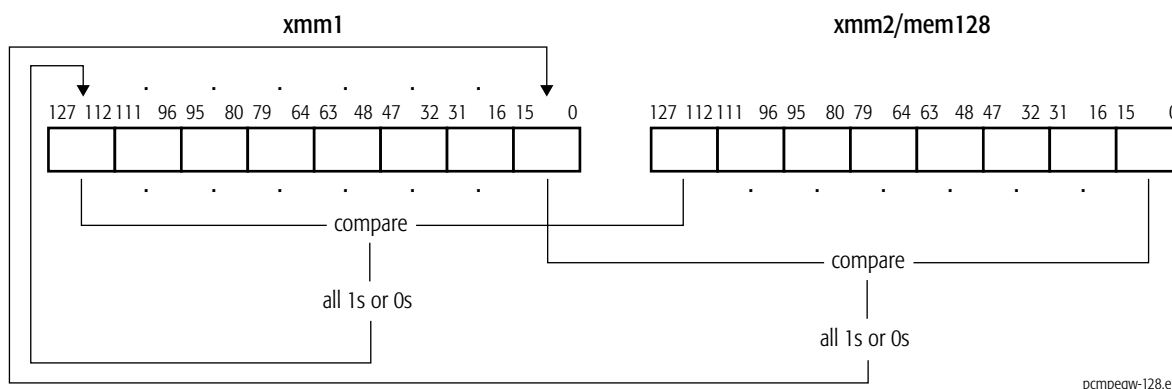
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPEQW**Packed Compare Equal Words**

The PCMPEQW instruction compares corresponding packed 16-bit values in the first and second source operands and writes the result of each comparison in the corresponding 16 bits of the destination (first source). For each pair of words, if the values are equal, the result is all 1s. If the values are not equal, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PCMPEQW <i>xmm1, xmm2/mem128</i>	66 0F 75 /r	Compares packed 16-bit values in an XMM register and an XMM register or 128-bit memory location.



pcmpeqw-128.eps

Related Instructions

PCMPEQB, PCMPEQD, PCMPGTB, PCMPGTD, PCMPGTW

rFLAGS Affected

None

MXCSR Flags Affected

None

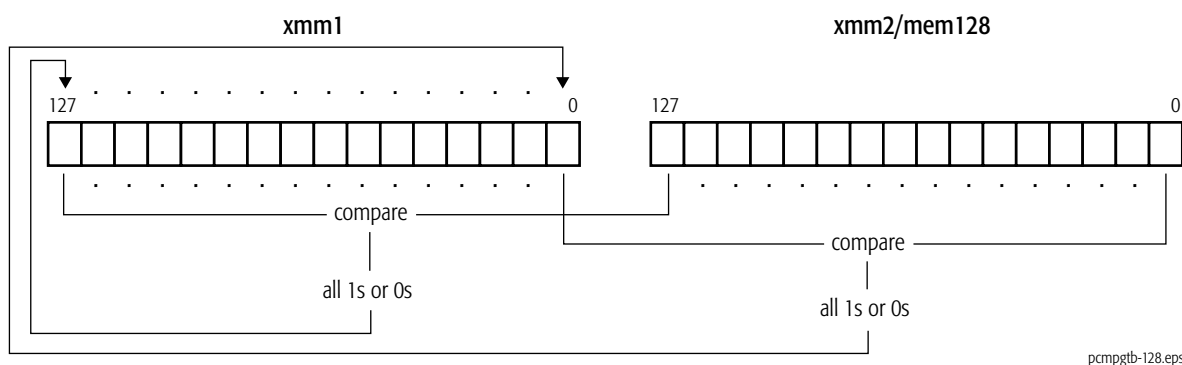
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPGTB**Packed Compare Greater Than Signed Bytes**

The PCMPGTB instruction compares corresponding packed signed bytes in the first and second source operands and writes the result of each comparison in the corresponding byte of the destination (first source). For each pair of bytes, if the value in the first source operand is greater than the value in the second source operand, the result is all 1s. If the value in the first source operand is less than or equal to the value in the second source operand, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PCMPGTB <i>xmm1, xmm2/mem128</i>	66 0F 64 /r	Compares packed signed bytes in an XMM register and an XMM register or 128-bit memory location.

**Related Instructions**

PCMPEQB, PCMPEQD, PCMPEQW, PCMPGTD, PCMPGTW

rFLAGS Affected

None

MXCSR Flags Affected

None

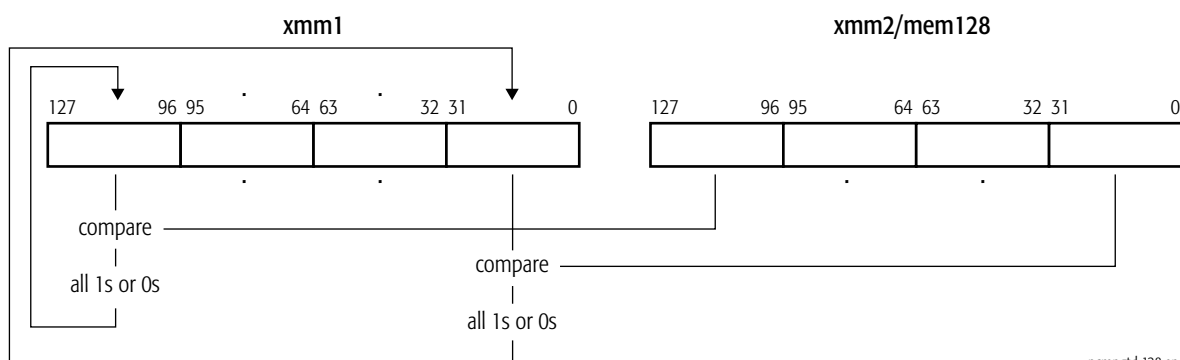
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPGTD**Packed Compare Greater Than Signed Doublewords**

The PCMPGTD instruction compares corresponding packed signed 32-bit values in the first and second source operands and writes the result of each comparison in the corresponding 32 bits of the destination (first source). For each pair of doublewords, if the value in the first source operand is greater than the value in the second source operand, the result is all 1s. If the value in the first source operand is less than or equal to the value in the second source operand, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PCMPGTD <i>xmm1, xmm2/mem128</i>	66 0F 66/r	Compares packed signed 32-bit values in an XMM register and an XMM register or 128-bit memory location.



pcmpgtd-128.eps

Related Instructions

PCMPEQB, PCMPEQD, PCMPEQW, PCMPGTB, PCMPGTW

rFLAGS Affected

None

MXCSR Flags Affected

None

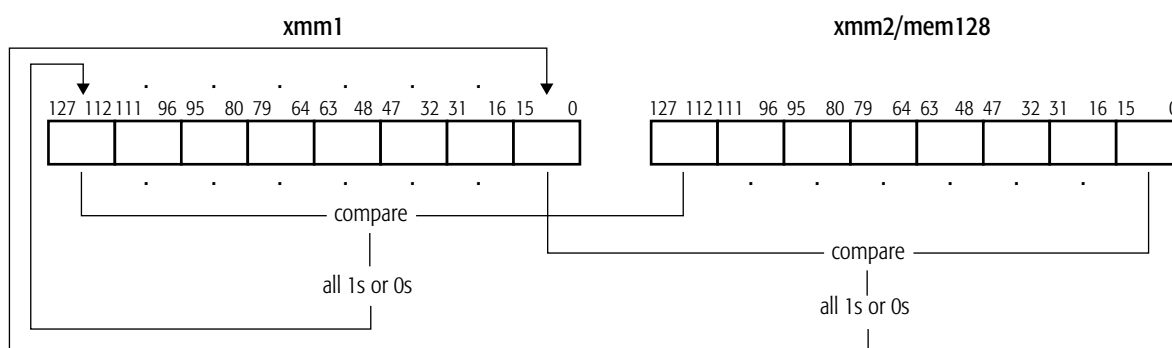
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPGTW**Packed Compare Greater Than Signed Words**

The PCMPGTW instruction compares corresponding packed signed 16-bit values in the first and second source operands and writes the result of each comparison in the corresponding 16 bits of the destination (first source). For each pair of words, if the value in the first source operand is greater than the value in the second source operand, the result is all 1s. If the value in the first source operand is less than or equal to the value in the second source operand, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PCMPGTW <i>xmm1, xmm2/mem128</i>	66 0F 65 /r	Compares packed signed 16-bit values in an XMM register and an XMM register or 128-bit memory location.



pcmpgtw-128.eps

Related Instructions

PCMPEQB, PCMPEQD, PCMPEQW, PCMPGTB, PCMPGTD

rFLAGS Affected

None

MXCSR Flags Affected

None

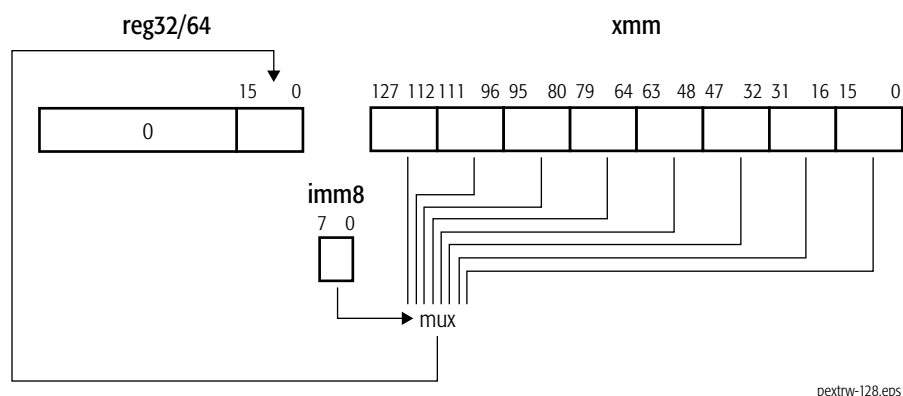
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PEXTRW Extract Packed Word

The PEXTRW instruction extracts a 16-bit value from an XMM register, as selected by the immediate byte operand (as shown in Table 1-2) and writes it to the low-order word of a 32-bit or 64-bit general-purpose register, with zero-extension to 32 or 64 bits.

Mnemonic	Opcode	Description
PEXTRW <i>reg32/64, xmm, imm8</i>	66 0F C5 /r <i>ib</i>	Extracts a 16-bit value from an XMM register and writes it to low-order 16 bits of a general-purpose register.



pextrw-128.eps

Table 1-2. Immediate-Byte Operand Encoding for 128-Bit PEXTRW

Immediate-Byte Bit Field	Value of Bit Field	Source Bits Extracted
2-0	0	15-0
	1	31-16
	2	47-32
	3	63-48
	4	79-64
	5	95-80
	6	111-96
	7	127-112

Related Instructions

PINSRW

rFLAGS Affected

None

MXCSR Flags Affected

None

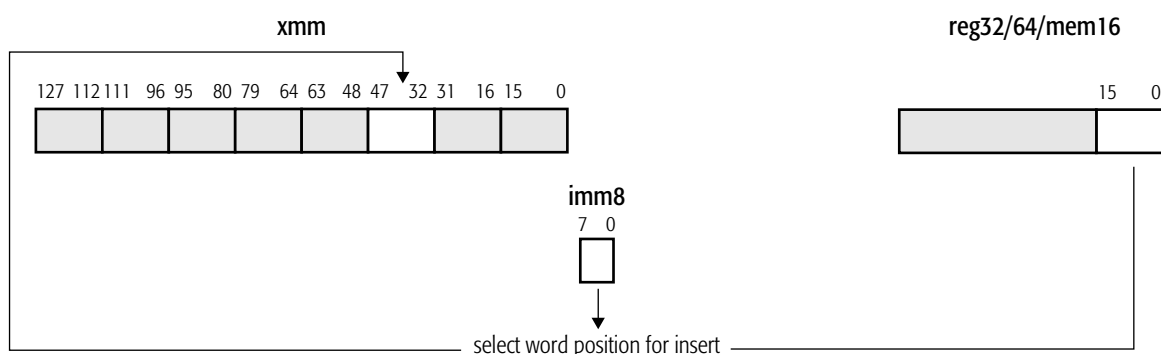
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PINSRW Packed Insert Word

The PINSRW instruction inserts a 16-bit value from the low-order word of a 32-bit or 64-bit general purpose register or a 16-bit memory location into an XMM register. The location in the destination register is selected by the immediate byte operand, as shown in Table 1-3. The other words in the destination register operand are not modified.

Mnemonic	Opcode	Description
PINSRW <i>xmm, reg32/64/mem16, imm8</i>	66 0F C4 /r ib	Inserts a 16-bit value from a general-purpose register or memory location into an XMM register.



pinsrw-128.eps

Table 1-3. Immediate-Byte Operand Encoding for 128-Bit PINSRW

Immediate-Byte Bit Field	Value of Bit Field	Destination Bits Filled
2–0	0	15–0
	1	31–16
	2	47–32
	3	63–48
	4	79–64
	5	95–80
	6	111–96
	7	127–112

Related Instructions

PEXTRW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

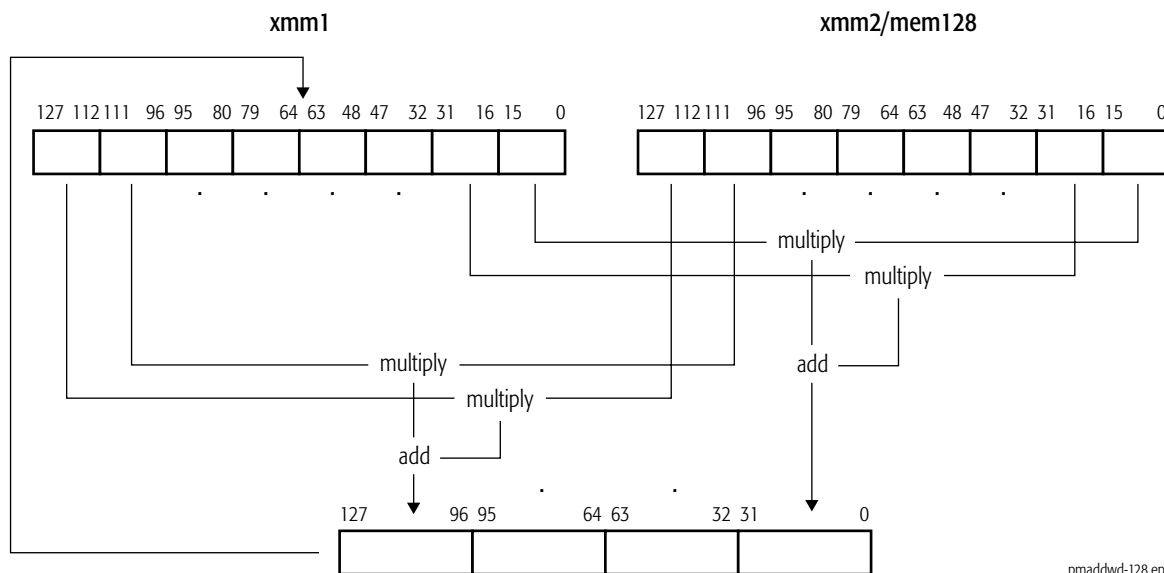
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)

Exception	Real	Virtual 8086	Protected	Cause of Exception
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

PMADDWD**Packed Multiply Words and Add Doublewords**

The PMADDWD instruction multiplies each packed 16-bit signed value in the first source operand by the corresponding packed 16-bit signed value in the second source operand, adds the adjacent intermediate 32-bit results of each multiplication (for example, the multiplication results for the adjacent bit fields 63–48 and 47–32, and 31–16 and 15–0), and writes the 32-bit result of each addition in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PMADDWD <i>xmm1, xmm2/mem128</i>	66 0F F5 /r	Multiplies eight packed 16-bit signed values in an XMM register and another XMM register or 128-bit memory location, adds intermediate results, and writes the result in the destination XMM register.



If all four of the 16-bit source operands used to produce a 32-bit multiply-add result have the value 8000h, the 32-bit result is 8000_0000h, which is incorrect.

Related Instructions

PMULHUW, PMULHW, PMULLW, PMULUDQ

rFLAGS Affected

None

MXCSR Flags Affected

None

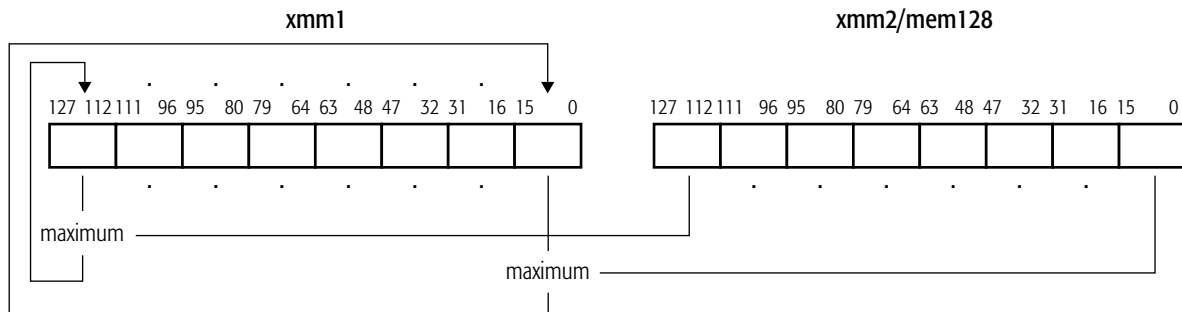
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMAXSW**Packed Maximum Signed Words**

The PMAXSW instruction compares each of the packed 16-bit signed integer values in the first source operand with the corresponding packed 16-bit signed integer value in the second source operand and writes the numerically greater of the two values for each comparison in the corresponding word of the destination (first source). The first source/destination and second source operands are an XMM register and an XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PMAXSW <i>xmm1, xmm2/mem128</i>	66 0F EE/ <i>r</i>	Compares packed signed 16-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the greater value of each comparison in destination XMM register.



pmaxsw-128.eps

Related Instructions

PMAXUB, PMINSW, PMINUB

rFLAGS Affected

None

MXCSR Flags Affected

None

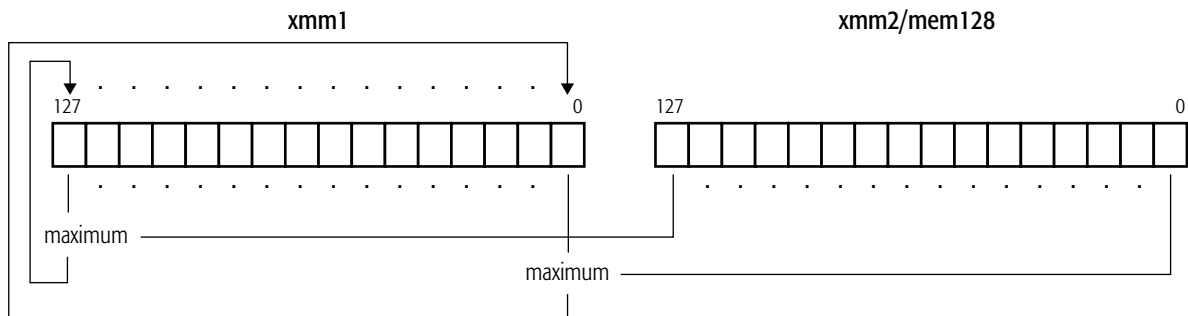
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	A memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMAXUB**Packed Maximum Unsigned Bytes**

The PMAXUB instruction compares each of the packed 8-bit unsigned integer values in the first source operand with the corresponding packed 8-bit unsigned integer value in the second source operand and writes the numerically greater of the two values for each comparison in the corresponding byte of the destination (first source). The first source/destination and second source operands are an XMM register and an XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PMAXUB <i>xmm1, xmm2/mem128</i>	66 0F DE/r	Compares packed unsigned 8-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the greater value of each compare in the destination XMM register.



pmaxub-128.eps

Related Instructions

PMAXSW, PMINSW, PMINUB

rFLAGS Affected

None

MXCSR Flags Affected

None

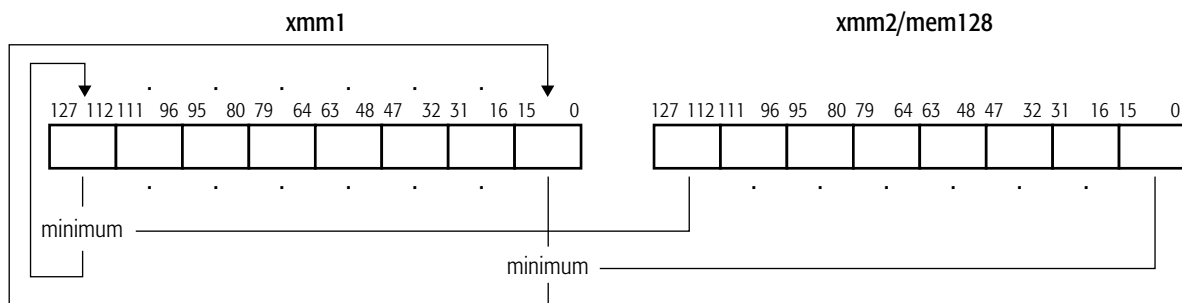
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	A memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMINSW**Packed Minimum Signed Words**

The PMINSW instruction compares each of the packed 16-bit signed integer values in the first source operand with the corresponding packed 16-bit signed integer value in the second source operand and writes the numerically lesser of the two values for each comparison in the corresponding word of the destination (first source). The first source/destination and second source operands are an XMM register and an XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PMINSW <i>xmm1, xmm2/mem128</i>	66 0F EA/r	Compares packed signed 16-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the lesser value of each compare in the destination XMM register.



pminsw-128.eps

Related Instructions

PMAWS, PMAUB, PMINUB

rFLAGS Affected

None

MXCSR Flags Affected

None

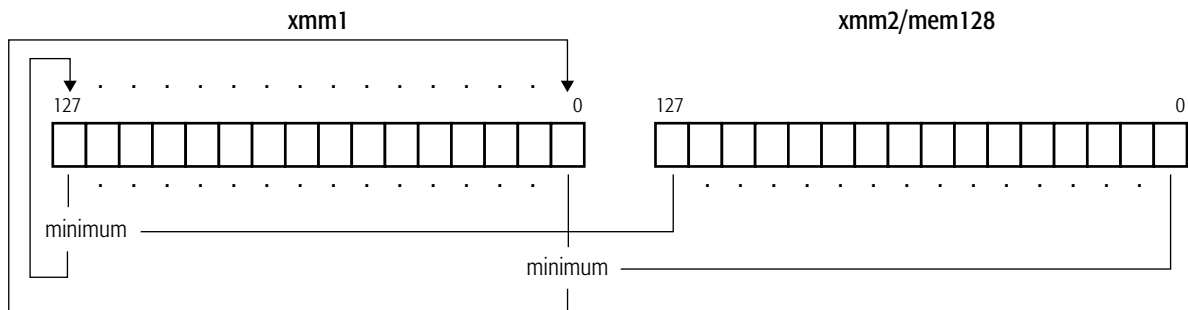
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	A memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMINUB**Packed Minimum Unsigned Bytes**

The PMINUB instruction compares each of the packed 8-bit unsigned integer values in the first source operand with the corresponding packed 8-bit unsigned integer value in the second source operand and writes the numerically lesser of the two values for each comparison in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PMINUB <i>xmm1, xmm2/mem128</i>	66 0F DA/r	Compares packed unsigned 8-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the lesser value of each comparison in the destination XMM register.



pminub-128.eps

Related Instructions

PMAWSW, PMAWSB, PMINSW

rFLAGS Affected

None

MXCSR Flags Affected

None

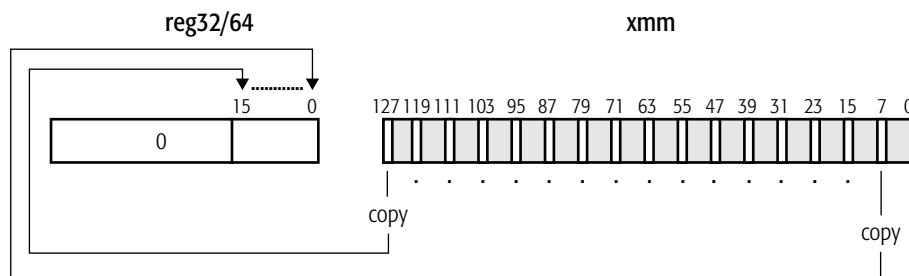
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	A memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMOVMSKB**Packed Move Mask Byte**

The PMOVMSKB instruction moves the most-significant bit of each byte in the source operand to the destination, with zero-extension to 32 or 64 bits. The destination and source operands are a 32-bit or 64-bit general-purpose register and an XMM register. The result is written to the low-order word of the general-purpose register.

Mnemonic	Opcode	Description
PMOVMSKB <i>reg32/64, xmm</i>	66 0F D7 /r	Moves most-significant bit of each byte in an XMM register to low-order word of a 32-bit or 64-bit general-purpose register.



pmovmskb-128.eps

Related Instructions

MOVMSKPD, MOVMSKPS

rFLAGS Affected

None

MXCSR Flags Affected

None

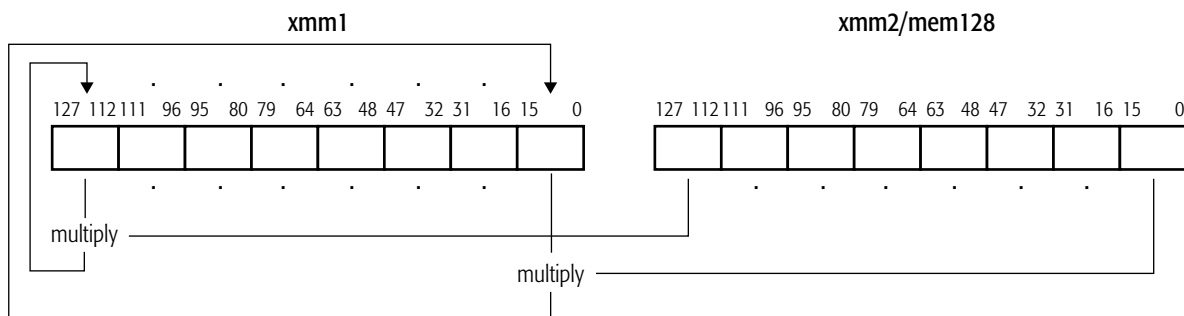
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.

PMULHUW**Packed Multiply High Unsigned Word**

The PMULHUW instruction multiplies each packed unsigned 16-bit values in the first source operand by the corresponding packed unsigned word in the second source operand and writes the high-order 16 bits of each intermediate 32-bit result in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PMULHUW <i>xmm1, xmm2/mem128</i>	66 0F E4 /r	Multiplies packed 16-bit values in an XMM register by the packed 16-bit values in another XMM register or 128-bit memory location and writes the high-order 16 bits of each result in the destination XMM register.



pmulhuw-128.eps

Related Instructions

PMADDWD, PMULHW, PMULLW, PMULUDQ

rFLAGS Affected

None

MXCSR Flags Affected

None

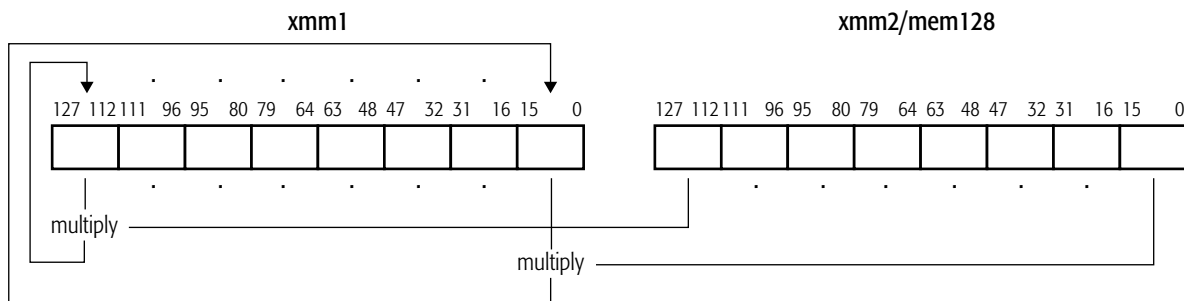
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMULHW**Packed Multiply High Signed Word**

The PMULHW instruction multiplies each packed 16-bit signed integer value in the first source operand by the corresponding packed 16-bit signed integer in the second source operand and writes the high-order 16 bits of the intermediate 32-bit result of each multiplication in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PMULHW <i>xmm1, xmm2/mem128</i>	66 0F E5 /r	Multiplies packed 16-bit signed integer values in an XMM register and another XMM register or 128-bit memory location and writes the high-order 16 bits of each result in the destination XMM register.



pmulhw-128.eps

Related Instructions

PMADDWD, PMULHUW, PMULLW, PMULUDQ

rFLAGS Affected

None

MXCSR Flags Affected

None

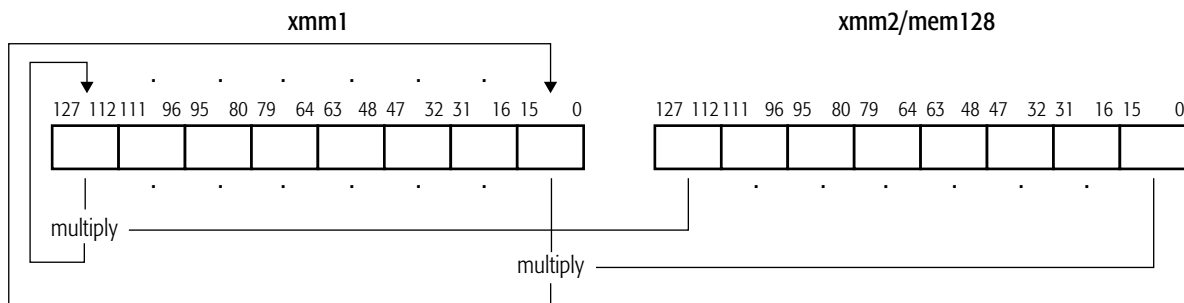
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMULLW**Packed Multiply Low Signed Word**

The PMULLW instruction multiplies each packed 16-bit signed integer value in the first source operand by the corresponding packed 16-bit signed integer in the second source operand and writes the low-order 16 bits of the intermediate 32-bit result of each multiplication in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PMULLW <i>xmm1, xmm2/mem128</i>	66 0F D5 /r	Multiplies packed 16-bit signed integer values in an XMM register and another XMM register or 128-bit memory location and writes the low-order 16 bits of each result in the destination XMM register.



pmullw-128.eps

Related Instructions

PMADDWD, PMULHUW, PMULHW, PMULUDQ

rFLAGS Affected

None

MXCSR Flags Affected

None

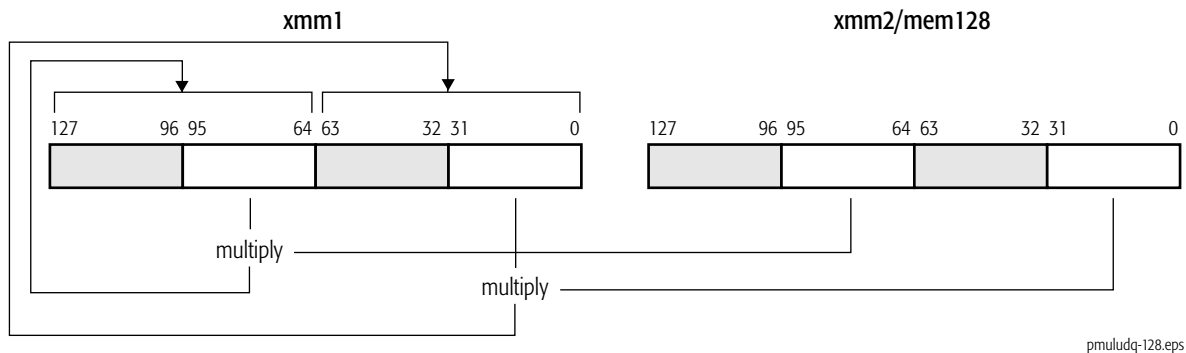
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMULUDQ**Packed Multiply Unsigned Doubleword and Store Quadword**

The PMULUDQ instruction multiplies two pairs of 32-bit unsigned integer values in the first and second source operands and writes the two 64-bit results in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location. The source operands are in the first (low-order) and third doublewords of the source operands, and the result of each multiply is stored in the first and second quadwords of the destination XMM register.

Mnemonic	Opcode	Description
PMULUDQ <i>xmm1, xmm2/mem128</i>	66 0F F4 /r	Multiplies two pairs of 32-bit unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes the two 64-bit results in the destination XMM register.

**Related Instructions**

PMADDWD, PMULHUW, PMULHW, PMULLW

rFLAGS Affected

None

MXCSR Flags Affected

None

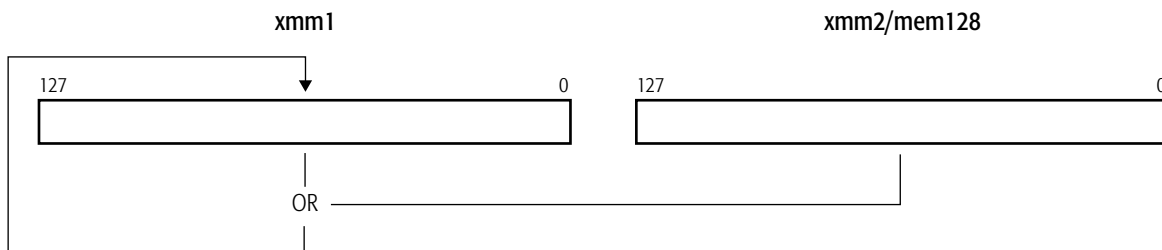
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

POR Packed Logical Bitwise OR

The POR instruction performs a bitwise logical OR of the values in the first and second source operands and writes the result in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
POR <i>xmm1, xmm2/mem128</i>	66 0F EB /r	Performs bitwise logical OR of values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



por-128.eps

Related Instructions

PAND, PANDN, PXOR

rFLAGS Affected

None

MXCSR Flags Affected

None

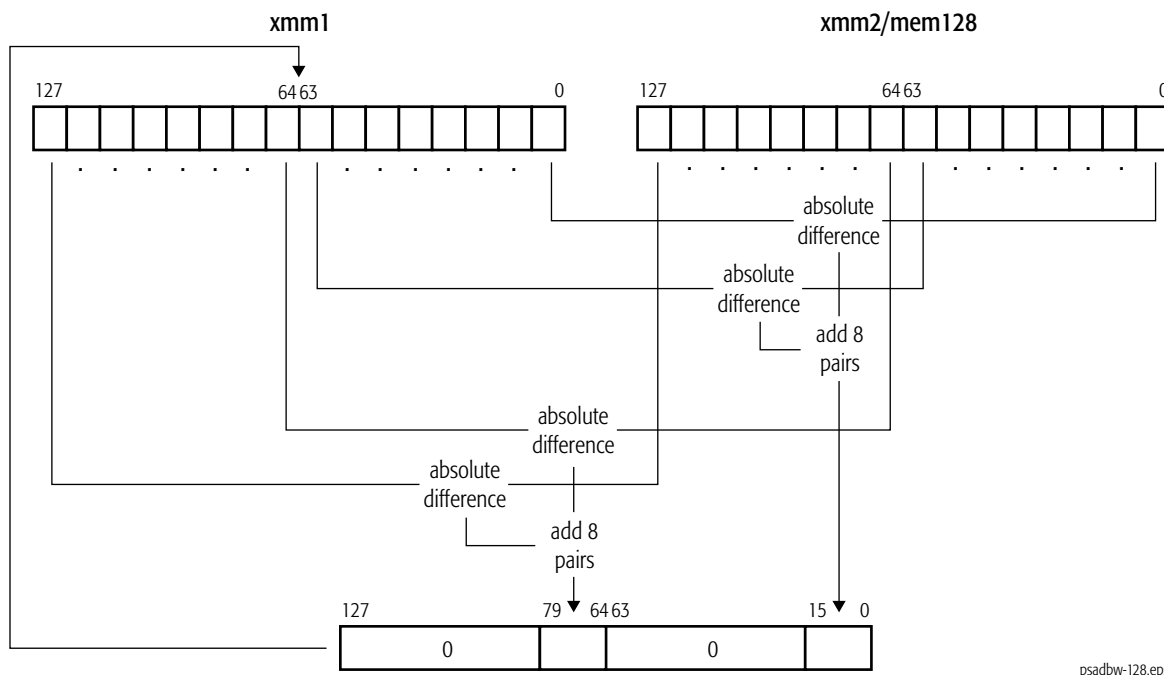
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSADBW**Packed Sum of Absolute Differences of Bytes Into a Word**

The PSADBW instruction computes the absolute differences of eight corresponding packed 8-bit unsigned integers in the first and second source operands and writes the unsigned 16-bit integer result of the sum of the eight differences in a word in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PSADBW <i>xmm1, xmm2/mem128</i>	66 0F F6 /r	Compute the sum of the absolute differences of two sets of packed 8-bit unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes the 16-bit unsigned integer result in the destination XMM register.



The sum of the differences of the eight bytes in the high-order quadwords of the source operands are written in the least-significant word of the high-order quadword in the destination XMM register, with the remaining bytes cleared to all 0s. The sum of

the differences of the eight bytes in the low-order quadwords of the source operands are written in the least-significant word of the low-order quadword in the destination XMM register, with the remaining bytes cleared to all 0s.

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSHUFD**Packed Shuffle Doublewords**

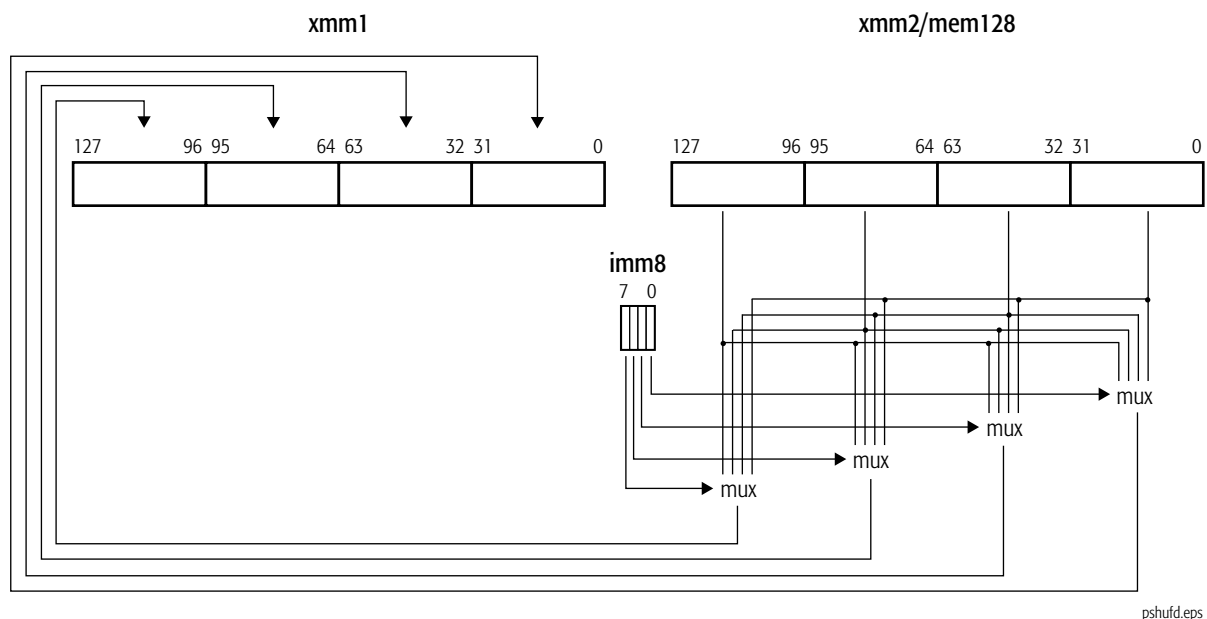
The PSHUFD instruction moves any one of the four packed doublewords in an XMM register or 128-bit memory location to each doubleword in another XMM register. In each case, the value of the destination doubleword is determined by a two-bit field in the immediate-byte operand, with bits 0 and 1 selecting the contents of the low-order doubleword, bits 2 and 3 selecting the second doubleword, bits 4 and 5 selecting the third doubleword, and bits 6 and 7 selecting the high-order doubleword. Refer to Table 1-4. A doubleword in the source operand may be copied to more than one doubleword in the destination.

Mnemonic**Opcode****Description**

PSHUFD *xmm1, xmm2/mem128, imm8*

66 0F 70 /r *ib*

Moves packed 32-bit values in an XMM register or 128-bit memory location to doubleword locations in another XMM register, as selected by the immediate-byte operand.



pshufd.eps

Table 1-4. Immediate-Byte Operand Encoding for PSHUFD

Immediate-Byte Bit Field	Value of Bit Field	Destination Bits Filled	Source Bits Moved
1–0	0	31–0	31–0
	1	31–0	63–32
	2	31–0	95–64
	3	31–0	127–96
3–2	0	63–32	31–0
	1	63–32	63–32
	2	63–32	95–64
	3	63–32	127–96
5–4	0	95–64	31–0
	1	95–64	63–32
	2	95–64	95–64
	3	95–64	127–96
7–6	0	127–96	31–0
	1	127–96	63–32
	2	127–96	95–64
	3	127–96	127–96

Related Instructions

PSHUFHW, PSHUFLW, PSHUFW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSHUFHW**Packed Shuffle High Words**

The PSHUFHW instruction moves any one of the four packed words in the high-order quadword of an XMM register or 128-bit memory location to each word in the high-order quadword of another XMM register. In each case, the value of the destination word is determined by a two-bit field in the immediate-byte operand, with bits 0 and 1 selecting the contents of the low-order word, bits 2 and 3 selecting the second word, bits 4 and 5 selecting the third word, and bits 6 and 7 selecting the high-order word. Refer to Table 1-5. A word in the source operand may be copied to more than one word in the destination. The low-order quadword of the source operand is copied to the low-order quadword of the destination register.

Mnemonic**Opcode****Description**

PSHUFHW *xmm1, xmm2/mem128, imm8*

F3 0F 70 /r ib

Shuffles packed 16-bit values in high-order quadword of an XMM register or 128-bit memory location and puts the result in high-order quadword of another XMM register.

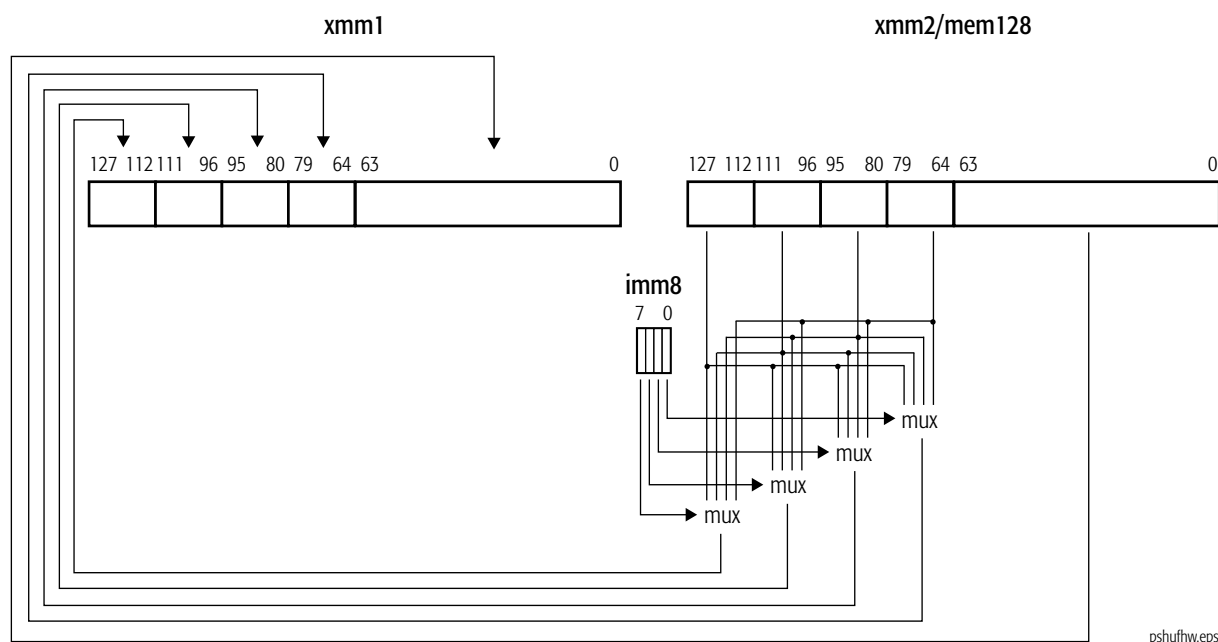


Table 1-5. Immediate-Byte Operand Encoding for PSHUFW

Immediate-Byte Bit Field	Value of Bit Field	Destination Bits Filled	Source Bits Moved
1–0	0	79–64	79–64
	1	79–64	95–80
	2	79–64	111–96
	3	79–64	127–112
3–2	0	95–80	79–64
	1	95–80	95–80
	2	95–80	111–96
	3	95–80	127–112
5–4	0	111–96	79–64
	1	111–96	95–80
	2	111–96	111–96
	3	111–96	127–112
7–6	0	127–112	79–64
	1	127–112	95–80
	2	127–112	111–96
	3	127–112	127–112

Related Instructions

PSHUFD, PSHUFLW, PSHUFW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSHUFLW**Packed Shuffle Low Words**

The PSHUFLW instruction moves any one of the four packed words in the low-order quadword of an XMM register or 128-bit memory location to each word in the low-order quadword of another XMM register. In each case, the selection of the value of the destination word is determined by a two-bit field in the immediate-byte operand, with bits 0 and 1 selecting the contents of the low-order word, bits 2 and 3 selecting the second word, bits 4 and 5 selecting the third word, and bits 6 and 7 selecting the high-order word. Refer to Table 1-6. A word in the source operand may be copied to more than one word in the destination. The high-order quadword of the source operand is copied to the high-order quadword of the destination register.

Mnemonic	Opcode	Description
PSHUFLW <i>xmm1, xmm2/mem128, imm8</i>	F2 0F 70 /r ib	Shuffles packed 16-bit values in low-order quadword of an XMM register or 128-bit memory location and puts the result in low-order quadword of another XMM register.

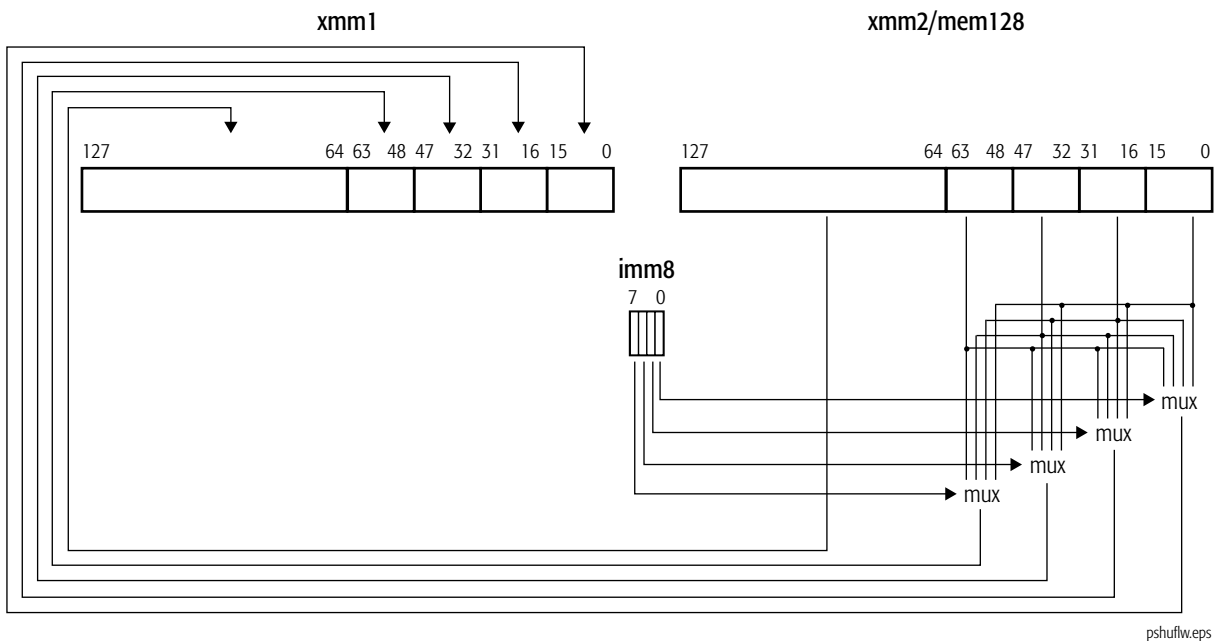


Table 1-6. Immediate-Byte Operand Encoding for PSHUFLW

Immediate-Byte Bit Field	Value of Bit Field	Destination Bits Filled	Source Bits Moved
1–0	0	15–0	15–0
	1	15–0	31–16
	2	15–0	47–32
	3	15–0	63–48
3–2	0	31–16	15–0
	1	31–16	31–16
	2	31–16	47–32
	3	31–16	63–48
5–4	0	47–32	15–0
	1	47–32	31–16
	2	47–32	47–32
	3	47–32	63–48
7–6	0	63–48	15–0
	1	63–48	31–16
	2	63–48	47–32
	3	63–48	63–48

Related Instructions

PSHUFD, PSFUFHW, PSFUFW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

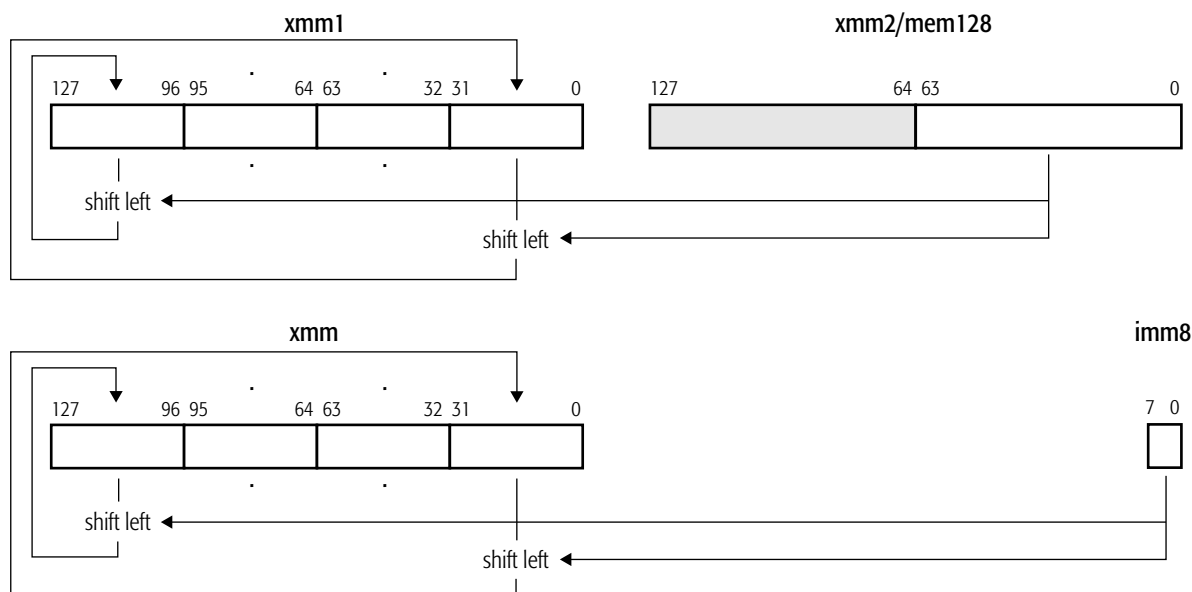
PSLLD**Packed Shift Left Logical Doublewords**

The PSLLD instruction left-shifts each of the packed 32-bit values in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding doubleword of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The low-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 31, the destination is cleared to all 0s.

Mnemonic	Opcode	Description
PSLLD <i>xmm1, xmm2/mem128</i>	66 0F F2 /r	Left-shifts packed doublewords in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSLLD <i>xmm, imm8</i>	66 0F 72 /6 ib	Left-shifts packed doublewords in an XMM register by the amount specified in an immediate byte value.



pslld-128.eps

Related Instructions

PSLLDQ, PSLQ, PSLW, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

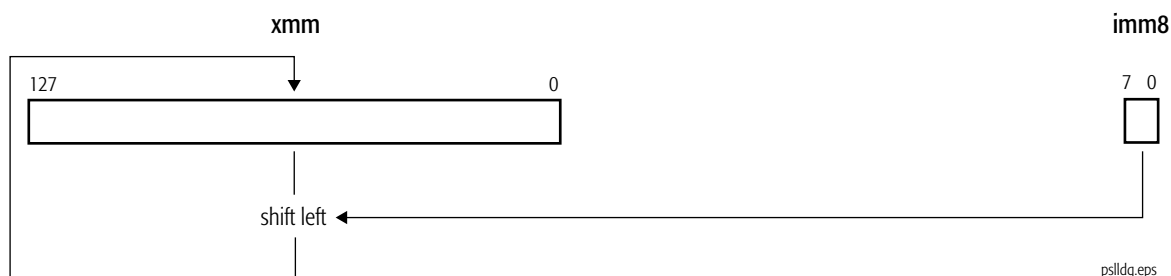
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSLLDQ**Packed Shift Left Logical Double Quadword**

The PSLLDQ left-shifts the 128-bit (double quadword) value in an XMM register by the number of bytes specified in an immediate byte value. The low-order bytes that are emptied by the shift operation are cleared to 0. If the shift value is greater than 15, the destination XMM register is cleared to all 0s.

Mnemonic	Opcode	Description
PSLLDQ <i>xmm, imm8</i>	66 0F 73 /7 <i>ib</i>	Left-shifts double quadword value in an XMM register by the amount specified in an immediate byte value.

**Related Instructions**

PSLLD, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.

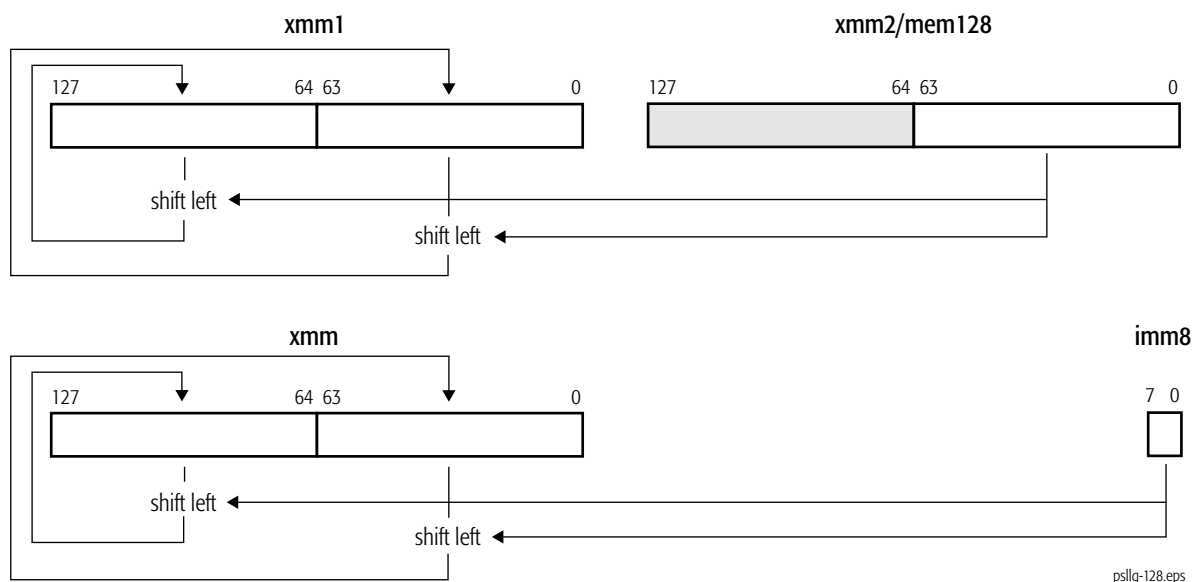
PSLLQ**Packed Shift Left Logical Quadwords**

The PSLLQ instruction left-shifts each 64-bit value in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding quadword of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The low-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 63, the destination is cleared to all 0s.

Mnemonic	Opcode	Description
PSLLQ <i>xmm1, xmm2/mem128</i>	66 0F F3 /r	Left-shifts packed quadwords in XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSLLQ <i>xmm, imm8</i>	66 0F 73 /6 ib	Left-shifts packed quadwords in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

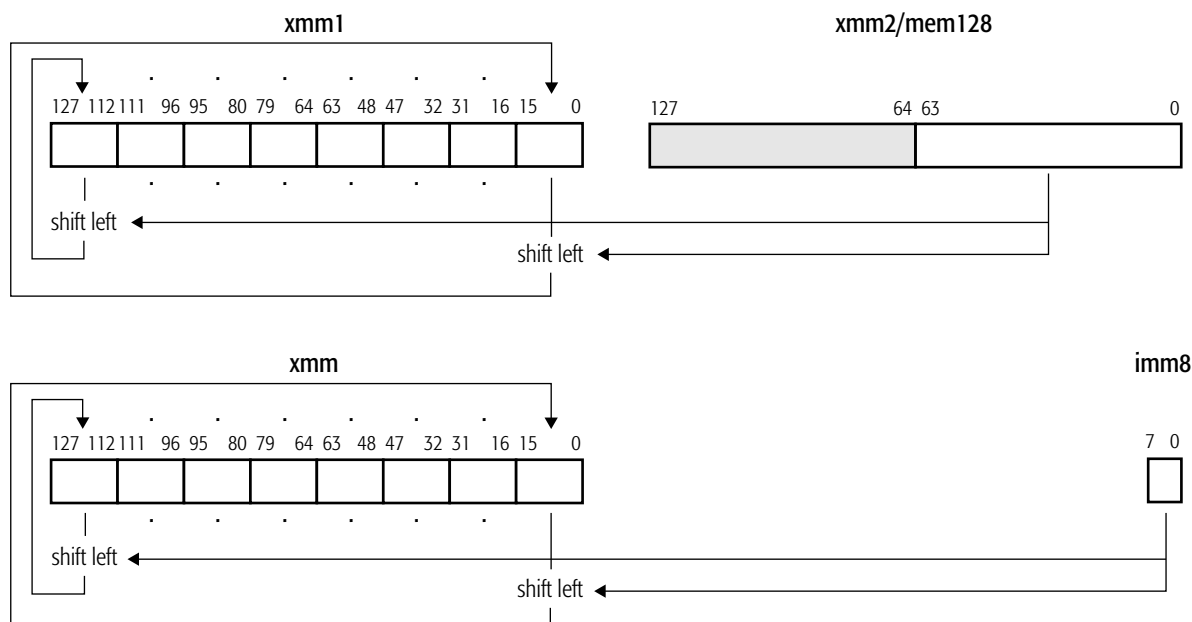
PSLLW**Packed Shift Left Logical Words**

The PSLLW instruction left-shifts each of the packed 16-bit values in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding word of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value

The low-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 15, the destination is cleared to all 0s.

Mnemonic	Opcode	Description
PSLLW <i>xmm1, xmm2/mem128</i>	66 0F F1 /r	Left-shifts packed words in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSLLW <i>xmm, imm8</i>	66 0F 71 /6 ib	Left-shifts packed words in an XMM register by the amount specified in an immediate byte value.



psllw-128.eps

Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

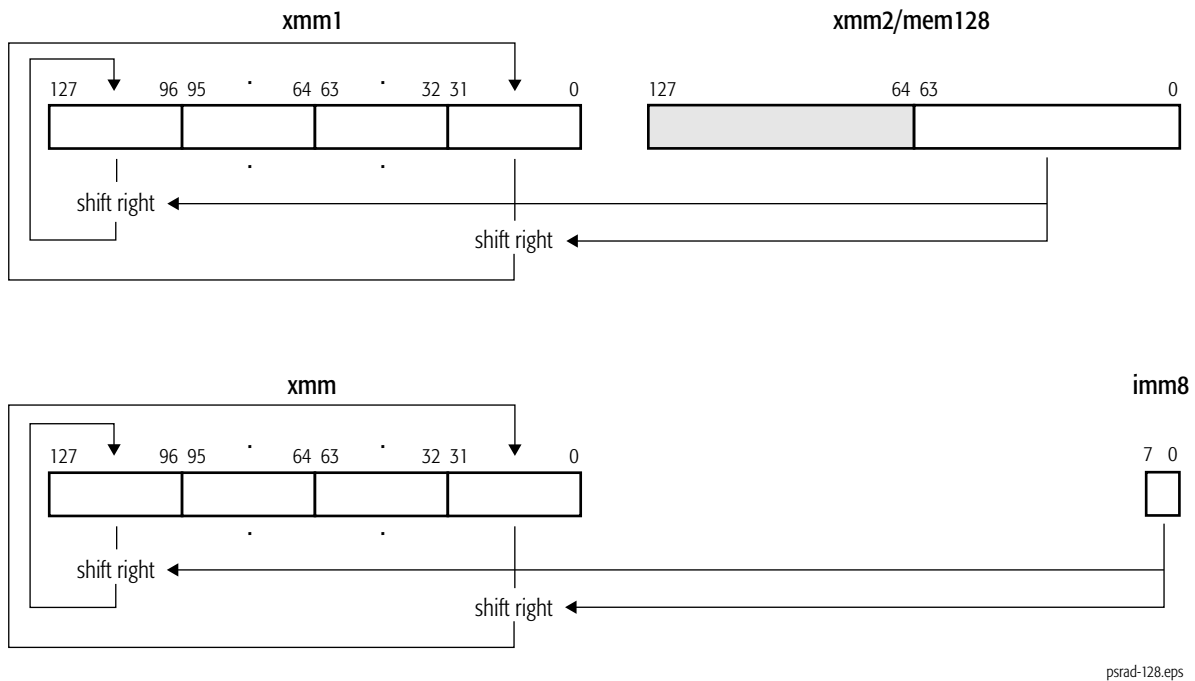
PSRAD**Packed Shift Right Arithmetic Doublewords**

The PSRAD instruction right-shifts each of the packed 32-bit values in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding doubleword of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The high-order bits that are emptied by the shift operation are filled with the sign bit of the doubleword's initial value. If the shift value is greater than 31, each doubleword in the destination is filled with the sign bit of the doubleword's initial value.

Mnemonic	Opcode	Description
PSRAD <i>xmm1, xmm2/mem128</i>	66 0F E2 /r	Right-shifts packed doublewords in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSRAD <i>xmm, imm8</i>	66 0F 72 /4 <i>ib</i>	Right-shifts packed doublewords in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAW, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

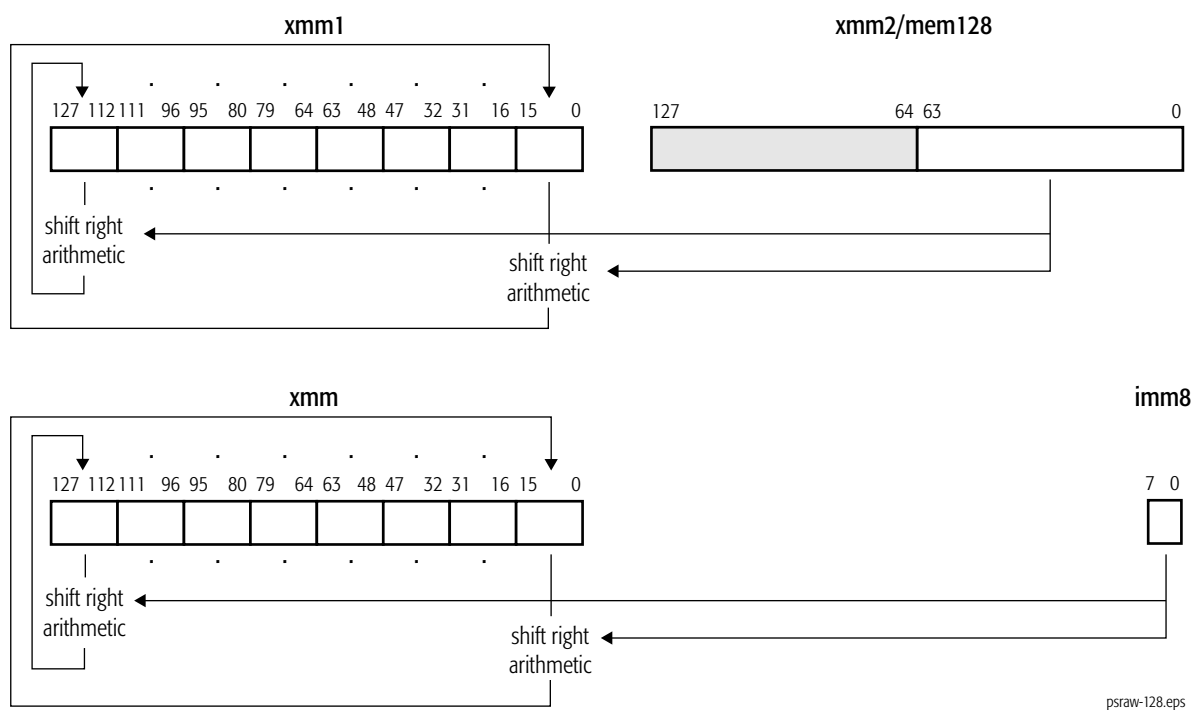
PSRAW**Packed Shift Right Arithmetic Words**

The PSRAW instruction right-shifts each of the packed 16-bit values in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding word of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The high-order bits that are emptied by the shift operation are filled with the sign bit of the word's initial value. If the shift value is greater than 15, each word in the destination is filled with the sign bit of the word's initial value.

Mnemonic	Opcode	Description
PSRAW <i>xmm1, xmm2/mem128</i>	66 0F E1 / <i>r</i>	Right-shifts packed words in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSRAW <i>xmm, imm8</i>	66 0F 71 /4 <i>ib</i>	Right-shifts packed words in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAD, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

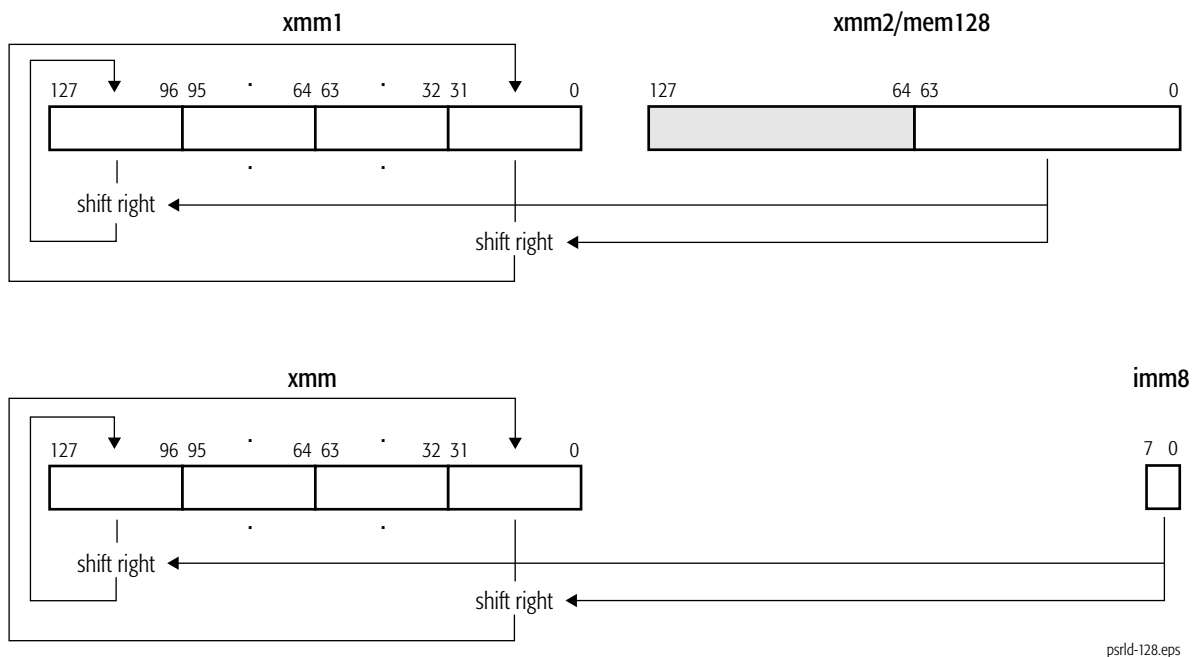
PSRLD**Packed Shift Right Logical Doublewords**

The PSRLD instruction right-shifts each of the packed 32-bit values in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding doubleword of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The high-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 31, the destination is cleared to 0.

Mnemonic	Opcode	Description
PSRLD <i>xmm1, xmm2/mem128</i>	66 0F D2 /r	Right-shifts packed doublewords in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSRLD <i>xmm, imm8</i>	66 0F 72 /2 ib	Right-shifts packed doublewords in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

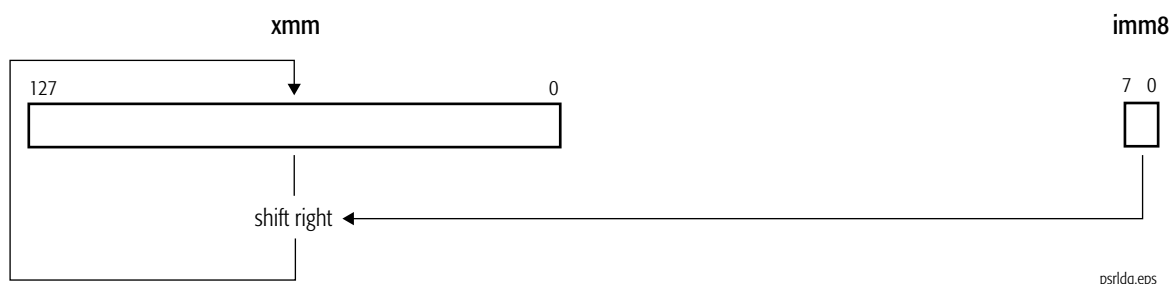
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSRLDQ Packed Shift Right Logical Double Quadword

The PSRLDQ right-shifts the 128-bit (double quadword) value in an XMM register by the number of bytes specified in an immediate byte value. The high-order bytes that are emptied by the shift operation are cleared to 0. If the shift value is greater than 15, the destination XMM register is cleared to all 0s.

Mnemonic	Opcode	Description
PSRLDQ <i>xmm, imm8</i>	66 0F 73 /3 <i>ib</i>	Right-shifts double quadword value in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.

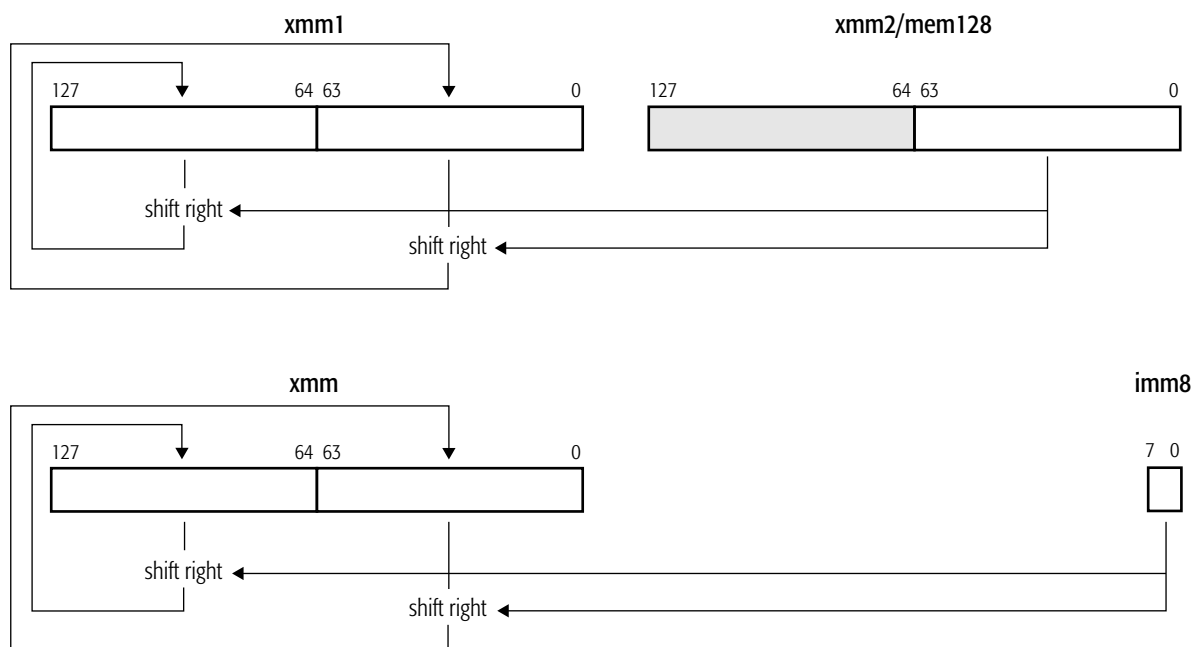
PSRLQ**Packed Shift Right Logical Quadwords**

The PSRLQ instruction right-shifts each 64-bit value in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding quadword of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The high-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 63, the destination is cleared to 0.

Mnemonic	Opcode	Description
PSRLQ <i>xmm1, xmm2/mem128</i>	66 0F D3 /r	Right-shifts packed quadwords in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSRLQ <i>xmm, imm8</i>	66 0F 73 /2 ib	Right-shifts packed quadwords in an XMM register by the amount specified in an immediate byte value.



psrlq-128.eps

Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

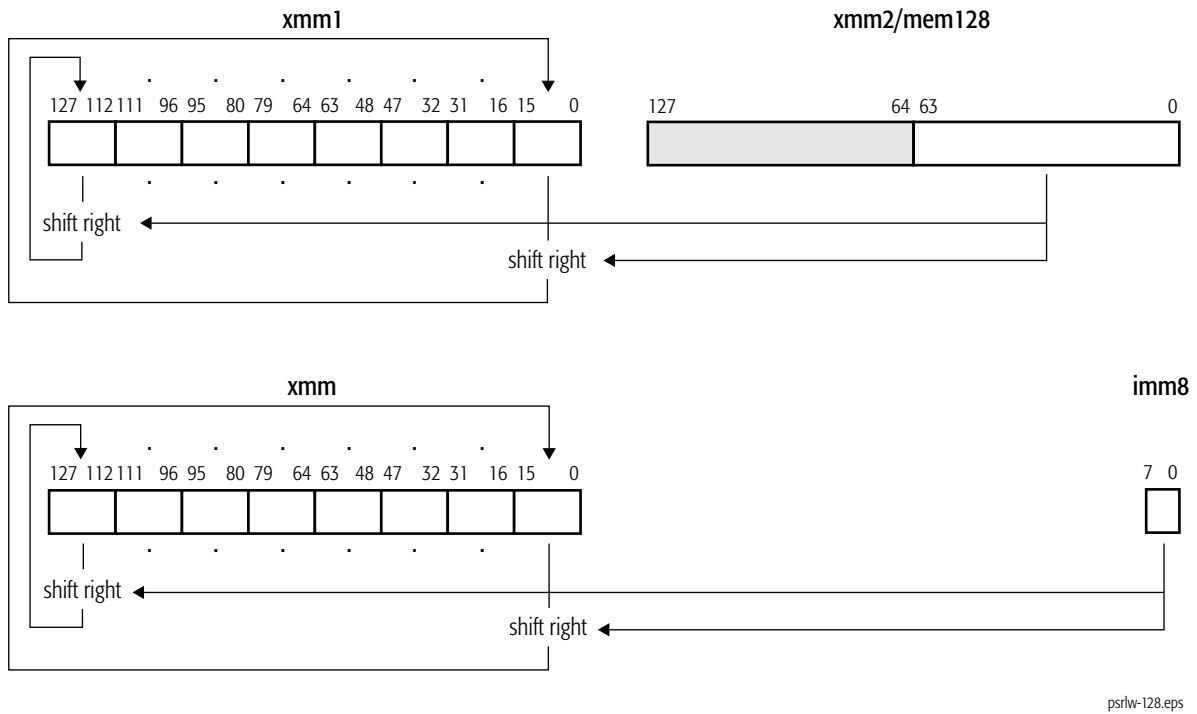
PSRLW**Packed Shift Right Logical Words**

The PSRLW instruction right-shifts each of the packed 16-bit values in the first source operand by the number of bits specified in the second operand and writes each shifted value in the corresponding word of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The high-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 15, the destination is cleared to 0.

Mnemonic	Opcode	Description
PSRLW <i>xmm1, xmm2/mem128</i>	66 0F D1 / <i>r</i>	Right-shifts packed words in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSRLW <i>xmm, imm8</i>	66 0F 71 /2 <i>ib</i>	Right-shifts packed words in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLQ

rFLAGS Affected

None

MXCSR Flags Affected

None

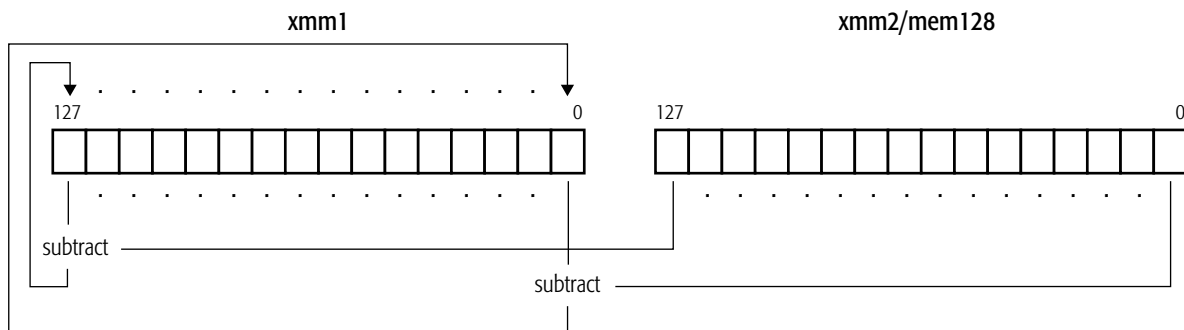
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBB**Packed Subtract Bytes**

The PSUBB instruction subtracts each packed 8-bit integer value in the second source operand from the corresponding packed 8-bit integer in the first source operand and writes the integer result of each subtraction in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PSUBB <i>xmm1, xmm2/mem128</i>	66 0F F8 /r	Subtracts packed byte integer values in an XMM register or 128-bit memory location from packed byte integer values in another XMM register and writes the result in the destination XMM register.



psubb-128.eps

This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 8 bits of each result are written in the destination.

Related Instructions

PSUBD, PSUBQ, PSUBSB, PSUBSW, PSUBUSB, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

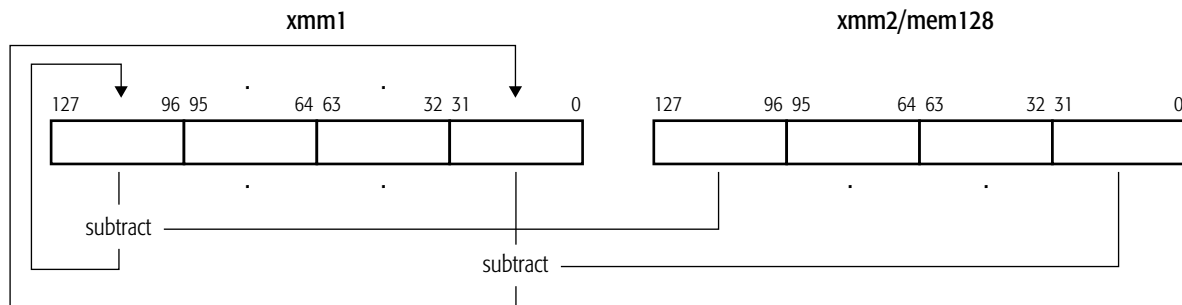
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBD**Packed Subtract Doublewords**

The PSUBD instruction subtracts each packed 32-bit integer value in the second source operand from the corresponding packed 32-bit integer in the first source operand and writes the integer result of each subtraction in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PSUBD <i>xmm1, xmm2/mem128</i>	66 0F FA/r	Subtracts packed 32-bit integer values in an XMM register or 128-bit memory location from packed 32-bit integer values in another XMM register and writes the result in the destination XMM register.



psubd-128.eps

This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 32 bits of each result are written in the destination.

Related Instructions

PSUBB, PSUBQ, PSUBSB, PSUBSW, PSUBUSB, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

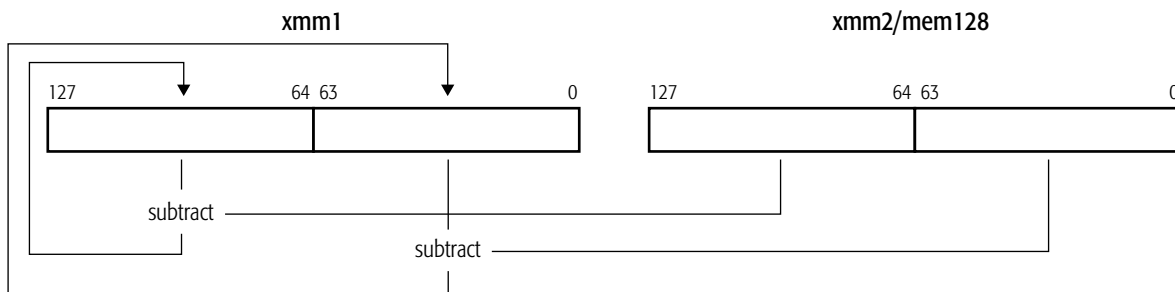
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBQ**Packed Subtract Quadword**

The PSUBQ instruction subtracts each packed 64-bit integer value in the second source operand from the corresponding packed 64-bit integer in the first source operand and writes the integer result of each subtraction in the corresponding quadword of the destination (first source). The first source/destination and source operands are an XMM register and another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PSUBQ <i>xmm1, xmm2/mem128</i>	66 0F FB /r	Subtracts packed 64-bit integer values in an XMM register or 128-bit memory location from packed 64-bit integer values in another XMM register and writes the result in the destination XMM register.



This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 64 bits of each result are written in the destination.

Related Instructions

PSUBB, PSUBD, PSUBSB, PSUBSW, PSUBUSB, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

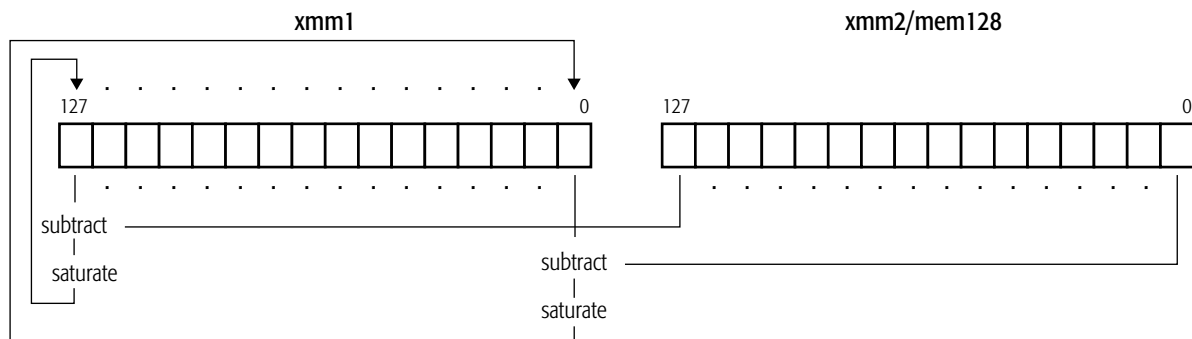
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBSB**Packed Subtract Signed With Saturation Bytes**

The PSUBSB instruction subtracts each packed 8-bit signed integer value in the second source operand from the corresponding packed 8-bit signed integer in the first source operand and writes the signed integer result of each subtraction in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PSUBSB <i>xmm1, xmm2/mem128</i>	66 0F E8 /r	Subtracts packed byte signed integer values in an XMM register or 128-bit memory location from packed byte integer values in another XMM register and writes the result in the destination XMM register.



psubsb-128.eps

For each packed value in the destination, if the value is larger than the largest signed 8-bit integer, it is saturated to 7Fh, and if the value is smaller than the smallest signed 8-bit integer, it is saturated to 80h.

Related Instructions

PSUBB, PSUBD, PSUBQ, PSUBSW, PSUBUSB, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

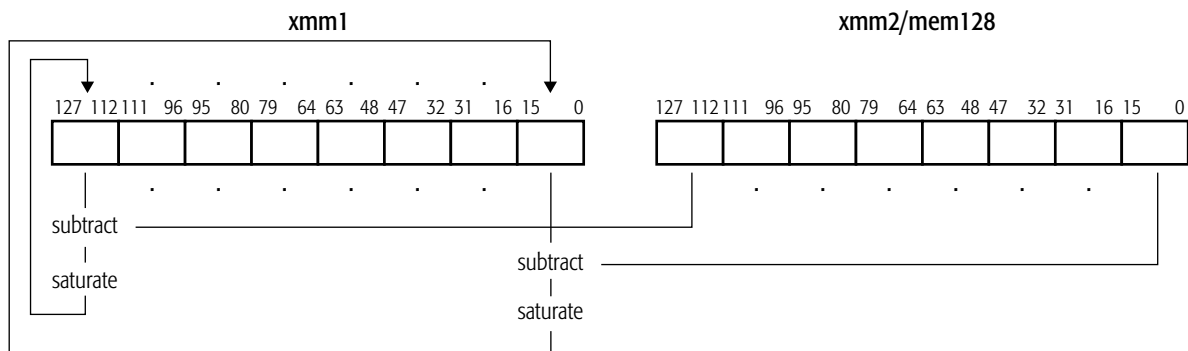
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBSW**Packed Subtract Signed With Saturation Words**

The PSUBSW instruction subtracts each packed 16-bit signed integer value in the second source operand from the corresponding packed 16-bit signed integer in the first source operand and writes the signed integer result of each subtraction in the corresponding word of the destination (first source). The first source/destination and source operands are an XMM register and another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PSUBSW <i>xmm1, xmm2/mem128</i>	66 0F E9 /r	Subtracts packed 16-bit signed integer values in an XMM register or 128-bit memory location from packed 16-bit integer values in another XMM register and writes the result in the destination XMM register.



psubsw-128.eps

For each packed value in the destination, if the value is larger than the largest signed 16-bit integer, it is saturated to 7FFFh, and if the value is smaller than the smallest signed 16-bit integer, it is saturated to 8000h.

Related Instructions

PSUBB, PSUBD, PSUBQ, PSUBSB, PSUBUSB, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

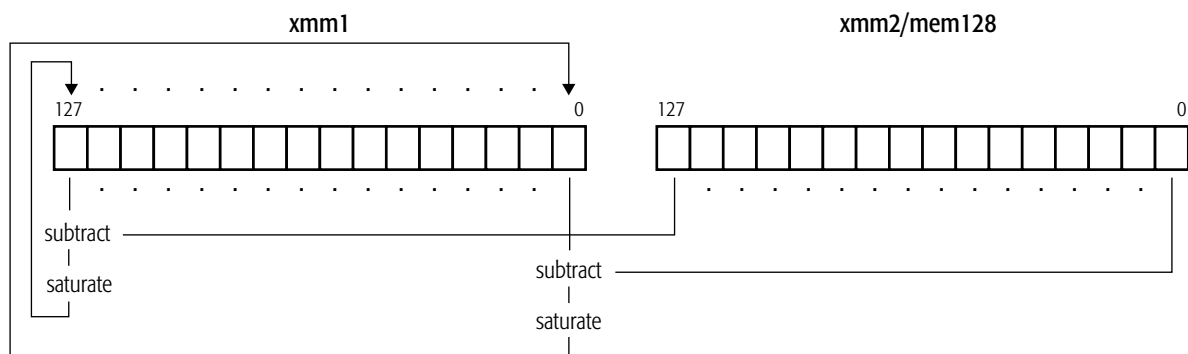
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBUSB**Packed Subtract Unsigned and Saturate Bytes**

The PSUBUSB instruction subtracts each packed 8-bit unsigned integer value in the second source operand from the corresponding packed 8-bit unsigned integer in the first source operand and writes the unsigned integer result of each subtraction in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PSUBUSB <i>xmm1, xmm2/mem128</i>	66 0F D8 /r	Subtracts packed byte unsigned integer values in an XMM register or 128-bit memory location from packed byte integer values in another XMM register and writes the result in the destination XMM register.



psubusb-128.eps

For each packed value in the destination, if the value is larger than the largest unsigned 8-bit integer, it is saturated to FFh, and if the value is smaller than the smallest unsigned 8-bit integer, it is saturated to 00h.

Related Instructions

PSUBB, PSUBD, PSUBQ, PSUBSB, PSUBSW, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

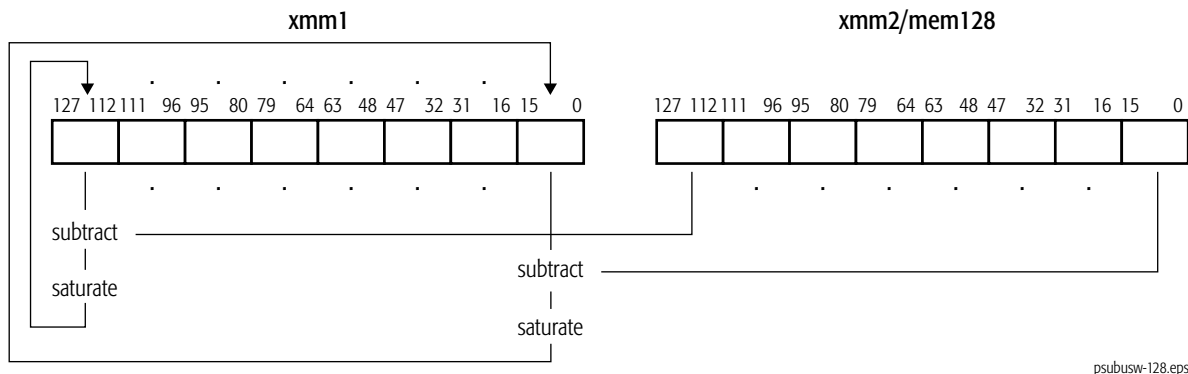
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBUSW**Packed Subtract Unsigned and Saturate Words**

The PSUBUSW instruction subtracts each packed 16-bit unsigned integer value in the second source operand from the corresponding packed 16-bit unsigned integer in the first source operand and writes the unsigned integer result of each subtraction in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PSUBUSW <i>xmm1, xmm2/mem128</i>	66 0F D9 /r	Subtracts packed 16-bit unsigned integer values in an XMM register or 128-bit memory location from packed 16-bit integer values in another XMM register and writes the result in the destination XMM register.



psubusw-128.eps

For each packed value in the destination, if the value is larger than the largest unsigned 16-bit integer, it is saturated to FFFFh, and if the value is smaller than the smallest unsigned 16-bit integer, it is saturated to 0000h.

Related Instructions

PSUBB, PSUBD, PSUBQ, PSUBSB, PSUBSW, PSUBUSB, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

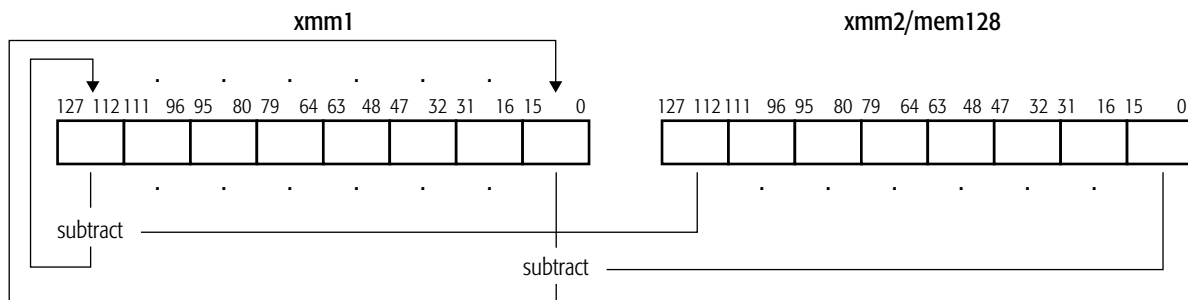
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBW**Packed Subtract Words**

The PSUBW instruction subtracts each packed 16-bit integer value in the second source operand from the corresponding packed 16-bit integer in the first source operand and writes the integer result of each subtraction in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PSUBW <i>xmm1, xmm2/mem128</i>	66 0F F9 /r	Subtracts packed 16-bit integer values in an XMM register or 128-bit memory location from packed 16-bit integer values in another XMM register and writes the result in the destination XMM register.



psubw-128.eps

This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 16 bits of the result are written in the destination.

Related Instructions

PSUBB, PSUBD, PSUBQ, PSUBSB, PSUBSW, PSUBUSB, PSUBUSW

rFLAGS Affected

None

MXCSR Flags Affected

None

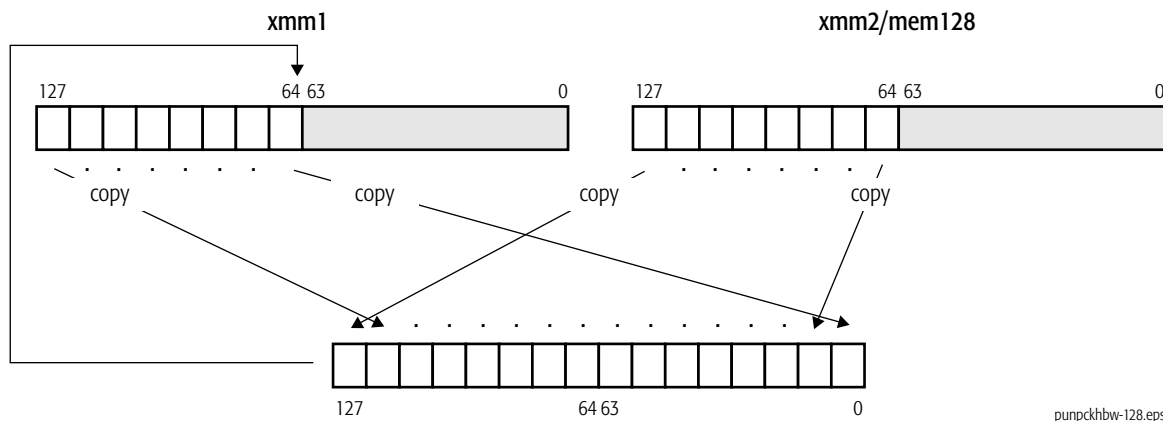
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PUNPCKHBW**Unpack and Interleave High Bytes**

The PUNPCKHBW instruction unpacks the high-order bytes from the first and second source operands and packs them into interleaved-byte words in the destination (first source). The low-order bytes of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PUNPCKHBW <i>xmm1, xmm2/mem128</i>	66 0F 68 /r	Unpacks the eight high-order bytes in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved bytes in the destination XMM register.



punpckhbw-128.eps

If the second source operand is all 0s, the destination contains the bytes from the first source operand zero-extended to 16 bits. This operation is useful for expanding unsigned 8-bit values to unsigned 16-bit operands for subsequent processing that requires higher precision.

Related Instructions

PUNPCKHDQ, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

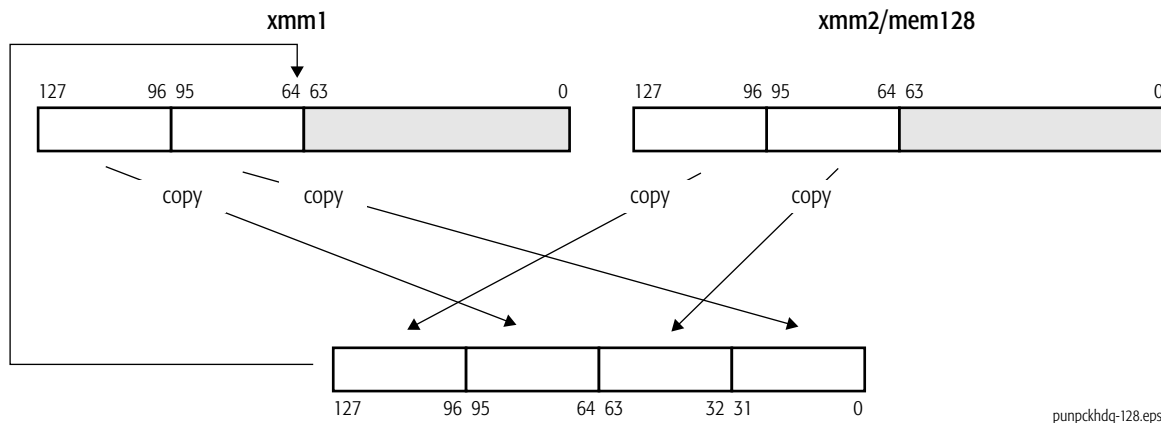
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PUNPCKHDQ**Unpack and Interleave High Doublewords**

The PUNPCKHDQ instruction unpacks the high-order doublewords from the first and second source operands and packs them into interleaved-doubleword quadwords in the destination (first source). The low-order doublewords of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PUNPCKHDQ <i>xmm1</i> , <i>xmm2/mem128</i>	66 0F 6A/ <i>r</i>	Unpacks two high-order doublewords in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved doublewords in the destination XMM register.



punpckhdq-128.eps

If the second source operand is all 0s, the destination contains the doubleword(s) from the first source operand zero-extended to 64 bits. This operation is useful for expanding unsigned 32-bit values to unsigned 64-bit operands for subsequent processing that requires higher precision.

Related Instructions

PUNPCKHBW, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PUNPCKHQDQ**Unpack and Interleave High Quadwords**

The PUNPCKHQDQ instruction unpacks the high-order quadwords from the first and second source operands and packs them into interleaved quadwords in the destination (first source). The first source/destination is an XMM register, and the second source operand is another XMM register or 128-bit memory location. The low-order quadwords of the source operands are ignored.

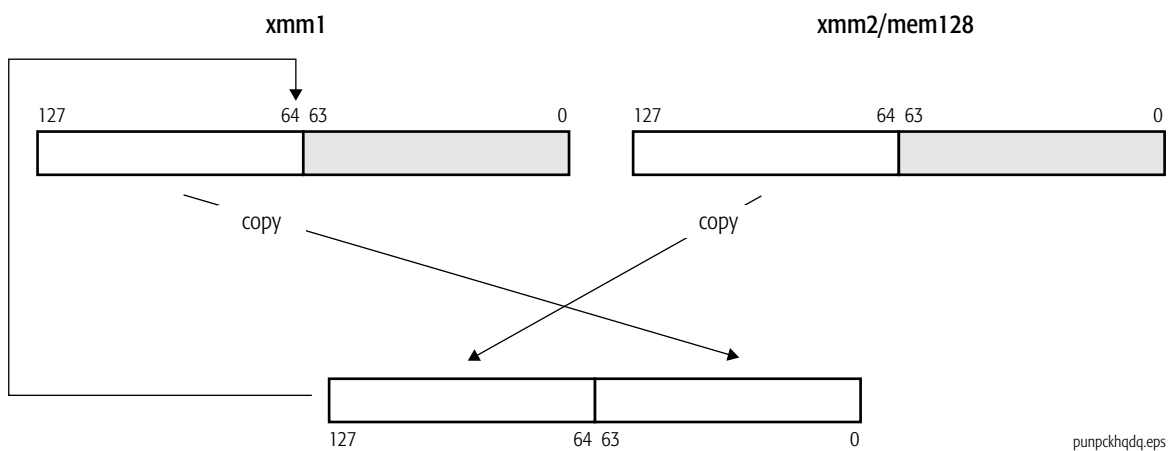
If the second source operand is all 0s, the destination contains the quadword from the first source operand zero-extended to 128 bits. This operation is useful for expanding unsigned 64-bit values to unsigned 128-bit operands for subsequent processing that requires higher precision.

Mnemonic**Opcode****Description**

PUNPCKHQDQ *xmm1, xmm2/mem128*

66 0F 6D /r

Unpacks high-order quadwords in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved quadwords in the destination XMM register.



punpckhqdq.eps

Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

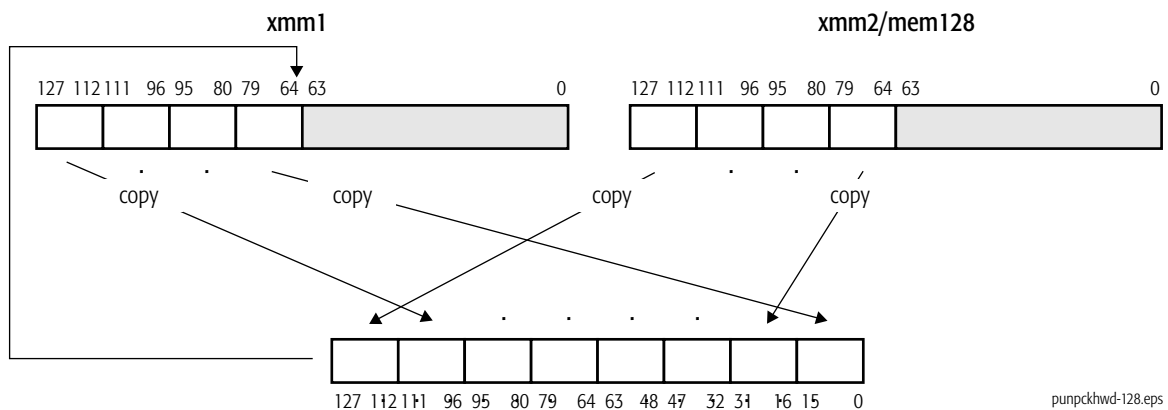
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PUNPCKHWD**Unpack and Interleave High Words**

The PUNPCKHWD instruction unpacks the high-order words from the first and second source operands and packs them into interleaved-word doublewords in the destination (first source). The low-order words of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PUNPCKHWD <i>xmm1, xmm2/mem128</i>	66 0F 69 /r	Unpacks four high-order words in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved words in the destination XMM register.



punpckhwd-128.eps

If the second source operand is all 0s, the destination contains the words from the first source operand zero-extended to 32 bits. This operation is useful for expanding unsigned 16-bit values to unsigned 32-bit operands for subsequent processing that requires higher precision.

Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHQDQ, PUNPCKLBW, PUNPCKLDQ, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

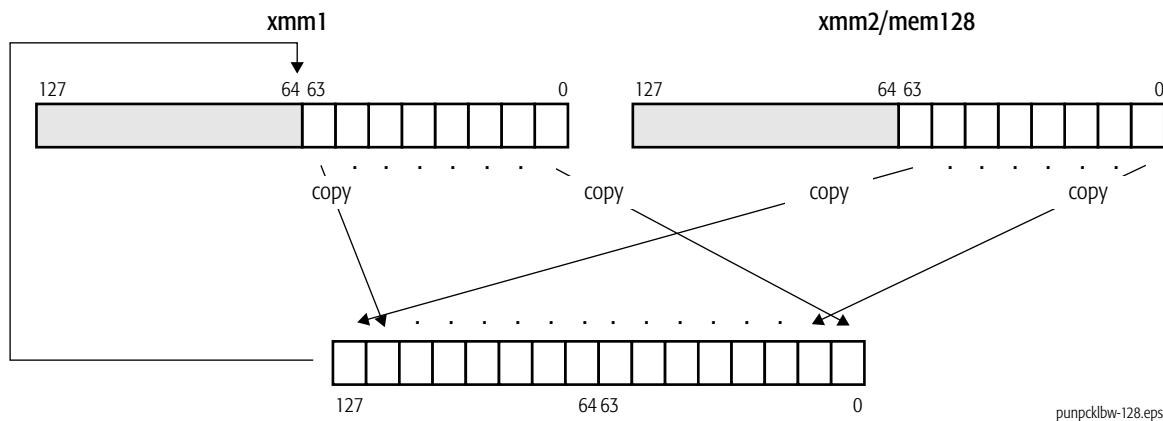
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PUNPCKLBW**Unpack and Interleave Low Bytes**

The PUNPCKLBW instruction unpacks the low-order bytes from the first and second source operands and packs them into interleaved-byte words in the destination (first source). The high-order bytes of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PUNPCKLBW <i>xmm1</i> , <i>xmm2/mem128</i>	66 0F 60 /r	Unpacks the eight low-order bytes in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved bytes in the destination XMM register.



If the second source operand is all 0s, the destination contains the bytes from the first source operand zero-extended to 16 bits. This operation is useful for expanding unsigned 8-bit values to unsigned 16-bit operands for subsequent processing that requires higher precision.

Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLDQ, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

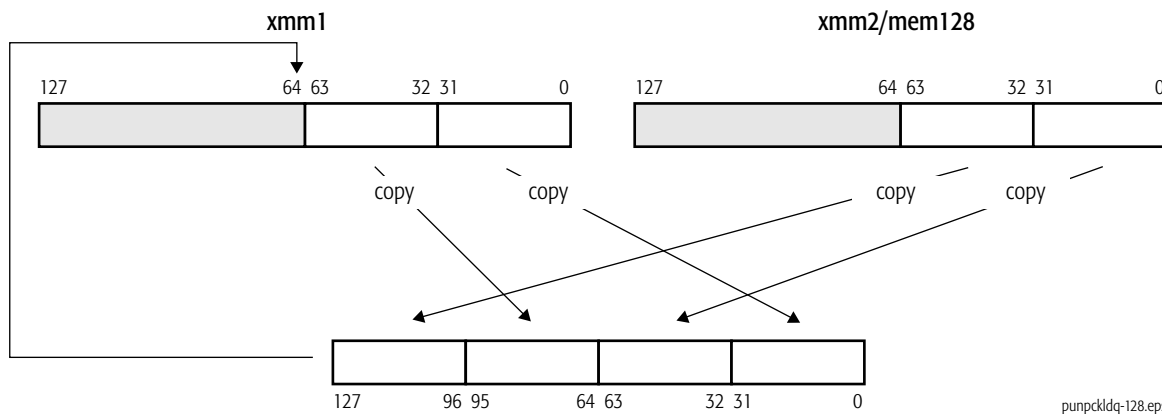
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PUNPCKLDQ**Unpack and Interleave Low Doublewords**

The PUNPCKLDQ instruction unpacks the low-order doublewords from the first and second source operands and packs them into interleaved-doubleword quadwords in the destination (first source). The high-order doublewords of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PUNPCKLDQ <i>xmm1, xmm2/mem128</i>	66 0F 62 /r	Unpacks two low-order doublewords in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved doublewords in the destination XMM register.



punpckldq-128.eps

If the second source operand is all 0s, the destination contains the doubleword(s) from the first source operand zero-extended to 64 bits. This operation is useful for expanding unsigned 32-bit values to unsigned 64-bit operands for subsequent processing that requires higher precision.

Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PUNPCKLQDQ**Unpack and Interleave Low Quadwords**

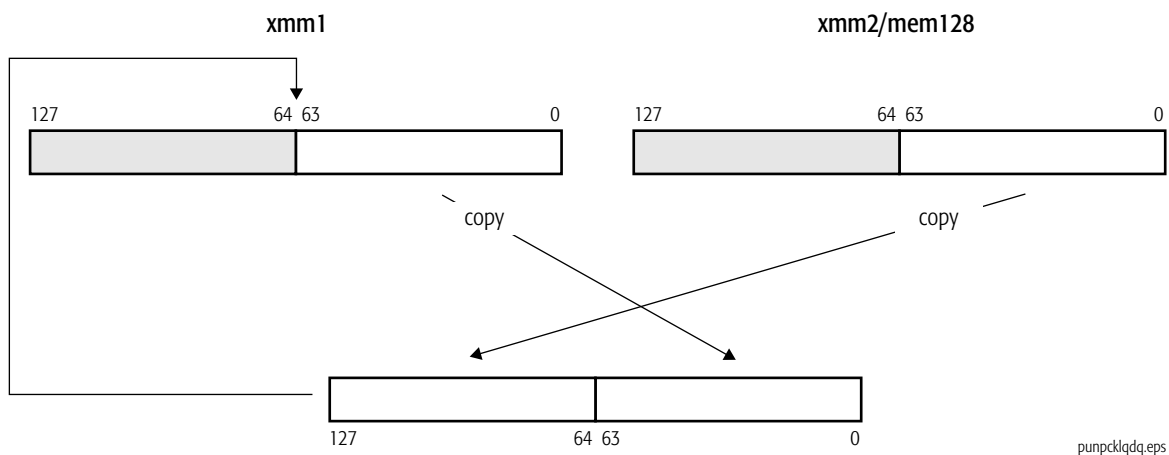
The PUNPCKLQDQ instruction unpacks the low-order quadwords from the first and second source operands and packs them into interleaved quadwords in the destination (first source). The first source/destination is an XMM register, and the second source operand is another XMM register or 128-bit memory location. The high-order quadwords of the source operands are ignored.

If the second source operand is all 0s, the destination contains the quadword from the first source operand zero-extended to 128 bits. This operation is useful for expanding unsigned 64-bit values to unsigned 128-bit operands for subsequent processing that requires higher precision.

Mnemonic**Opcode****Description**PUNPCKLQDQ *xmm1, xmm2/mem128*

66 0F 6C /r

Unpacks low-order quadwords in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved quadwords in the destination XMM register.



punpcklqdq.eps

Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

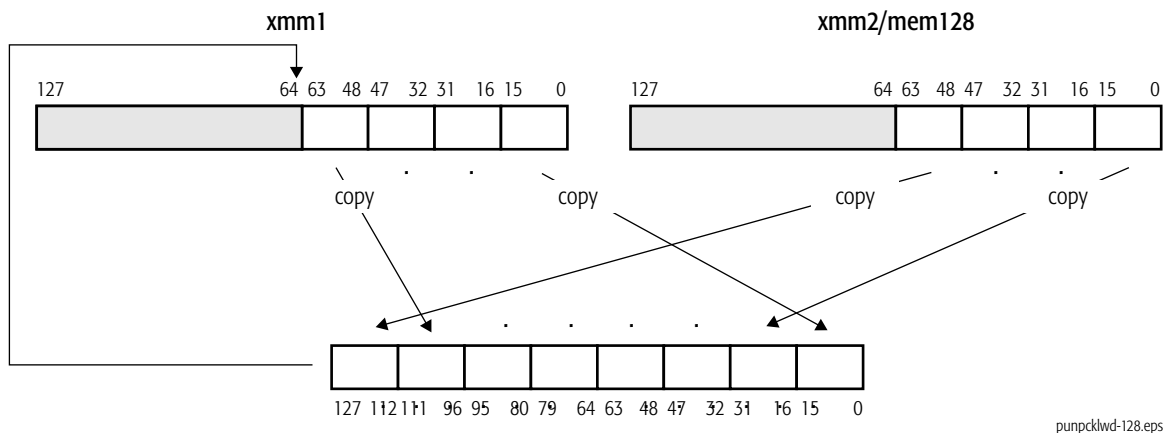
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PUNPCKLWD**Unpack and Interleave Low Words**

The PUNPCKLWD instruction unpacks the low-order words from the first and second source operands and packs them into interleaved-word doublewords in the destination (first source). The high-order words of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PUNPCKLWD <i>xmm1, xmm2/mem128</i>	66 0F 61 /r	Unpacks the four low-order words in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved words in the destination XMM register.



punpcklwd-128.eps

If the second source operand is all 0s, the destination contains the words from the first source operand zero-extended to 32 bits. This operation is useful for expanding unsigned 16-bit values to unsigned 32-bit operands for subsequent processing that requires higher precision.

Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLQDQ

rFLAGS Affected

None

MXCSR Flags Affected

None

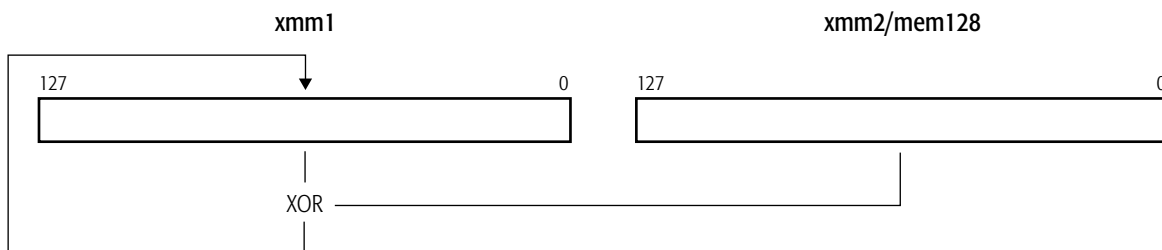
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PXOR**Packed Logical Bitwise Exclusive OR**

The PXOR instruction performs a bitwise exclusive OR of the values in the first and second source operands and writes the result in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
PXOR <i>xmm1, xmm2/mem128</i>	66 0F EF /r	Performs bitwise logical XOR of values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



pxor-128.eps

Related Instructions

PAND, PANDN, POR

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

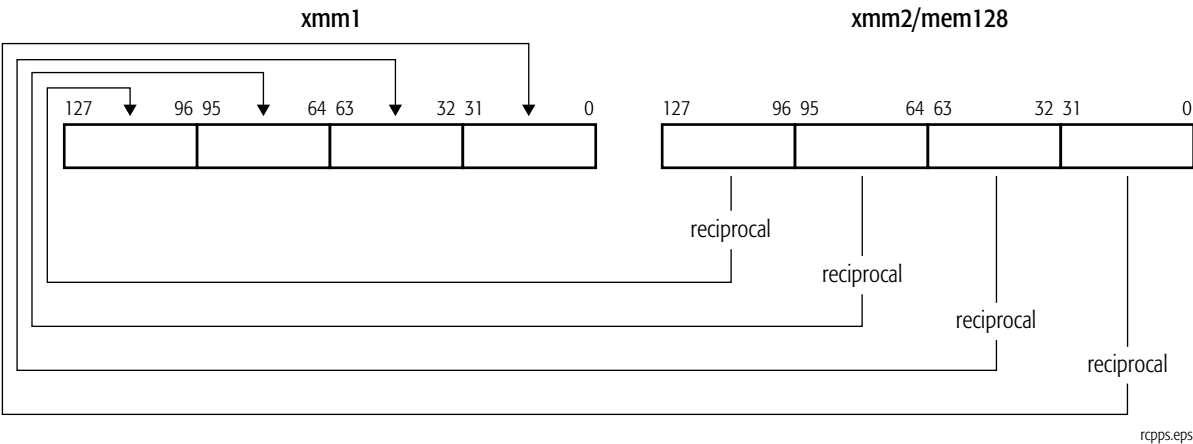
RCPPS

Reciprocal Packed Single-Precision Floating-Point

The RCPPS instruction computes the approximate reciprocal of each of the four packed single-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the corresponding doubleword of another XMM register. The rounding control bits (RC) in the MXCSR register have no effect on the result.

The maximum relative error is less than or equal to 1.5×2^{-12} . A source value of 0.0 returns an infinity of the source value’s sign. Denormal source operands are treated as signed 0.0. For negative source values other than 0.0, the QNaN floating-point indefinite value (“Indefinite Values” in Volume 1) is returned. Results that underflow are changed to signed 0.0. For both SNaN and QNaN source operands, a QNaN is returned.

Mnemonic	Opcode	Description
RCPPS <i>xmm1, xmm2/mem128</i>	OF 53 /r	Computes reciprocals of packed single-precision floating-point values in an XMM register or 128-bit memory location and writes result in the destination XMM register.



Related Instructions

RCPSS, RSQRTPS, RSQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

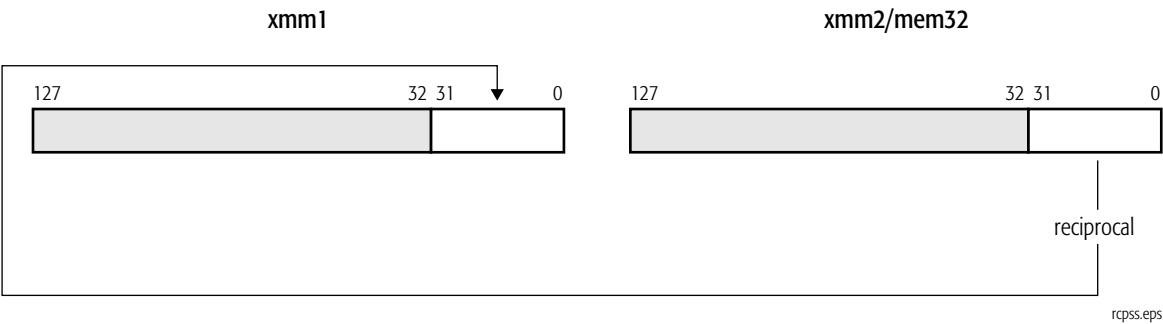
RCPSS

Reciprocal Scalar Single-Precision Floating-Point

The RCPSS instruction computes the approximate reciprocal of the low-order single-precision floating-point value in an XMM register or in a 32-bit memory location and writes the result in the low-order doubleword of another XMM register. The three high-order doublewords in the destination XMM register are not modified. The rounding control bits (RC) in the MXCSR register have no effect on the result.

The maximum relative error is less than or equal to 1.5×2^{-12} . A source value of 0.0 returns an infinity of the source value’s sign. Denormal source operands are treated as signed 0.0. For negative source values other than 0.0, the QNaN floating-point indefinite value (“Indefinite Values” in Volume 1) is returned. Results that underflow are changed to signed 0.0. For both SNaN and QNaN source operands, a QNaN is returned.

Mnemonic	Opcode	Description
RCPSS <i>xmm1, xmm2/mem32</i>	F3 0F 53 /r	Computes reciprocal of scalar single-precision floating-point value in an XMM register or 32-bit memory location and writes the result in the destination XMM register.



Related Instructions

RCPPS, RSQRTPS, RSQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

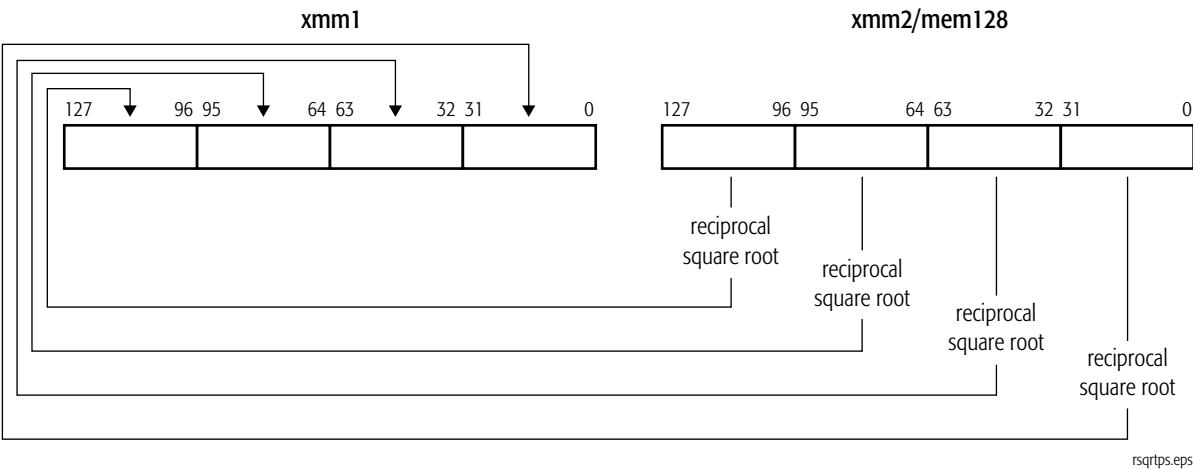
RSQRTPS

Reciprocal Square Root Packed Single-Precision Floating-Point

The RSQRTPS instruction computes the approximate reciprocal of the square root of each of the four packed single-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the corresponding doubleword of another XMM register. The rounding control bits (RC) in the MXCSR register have no effect on the result.

The maximum relative error for the approximate reciprocal is less than or equal to 1.5×2^{-12} . A source value of 0.0 returns an infinity of the source value’s sign. Denormal source operands are treated as signed 0.0. For negative source values other than 0.0, the QNaN floating-point indefinite value (“Indefinite Values” in Volume 1) is returned. For both SNaN and QNaN source operands, a QNaN is returned.

Mnemonic	Opcode	Description
RSQRTPS <i>xmm1, xmm2/mem128</i>	0F 52 /r	Computes reciprocals of square roots of packed single-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

RSQRTSS, SQRTPD, SQRTPS, SQRTSD, SQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

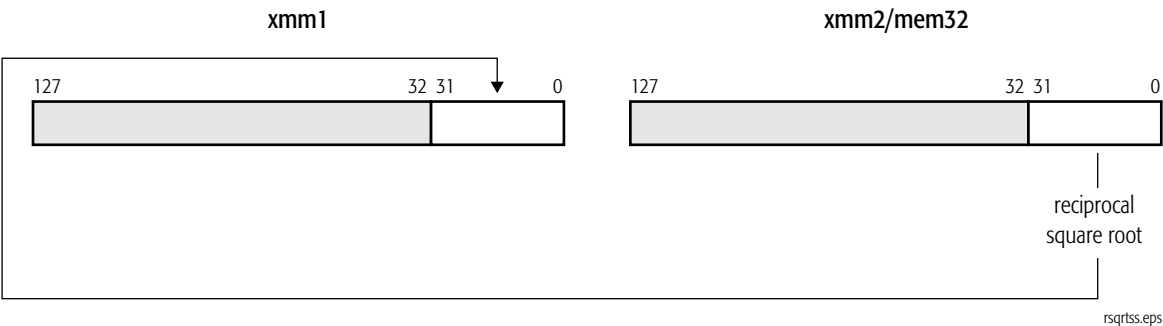
RSQRTSS

Reciprocal Square Root Scalar Single-Precision Floating-Point

The RSQRTSS instruction computes the approximate reciprocal of the square root of the low-order single-precision floating-point value in an XMM register or in a 32-bit memory location and writes the result in the low-order doubleword of another XMM register. The three high-order doublewords in the destination XMM register are not modified. The rounding control bits (RC) in the MXCSR register have no effect on the result.

The maximum relative error for the approximate reciprocal is less than or equal to 1.5×2^{-12} . A source value of 0.0 returns an infinity of the source value’s sign. Denormal source operands are treated as signed 0.0. For negative source values other than 0.0, the QNaN floating-point indefinite value (“Indefinite Values” in Volume 1) is returned. For both SNaN and QNaN source operands, a QNaN is returned.

Mnemonic	Opcode	Description
RSQRTPS <i>xmm1, xmm2/mem32</i>	F3 0F 52/r	Computes reciprocal of square root of single-precision floating-point value in an XMM register or 32-bit memory location and writes the result in the destination XMM register.



Related Instructions

RSQRTPS, SQRTPD, SQRTPS, SQRTSD, SQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

SHUFPD

Shuffle Packed Double-Precision Floating-Point

The SHUFPD instruction moves either of the two packed double-precision floating-point values in the first source operand to the low-order quadword of the destination (first source) and moves either of the two packed double-precision floating-point values in the second source operand to the high-order quadword of the destination. In each case, the value of the destination quadword is determined by the least-significant two bits in the immediate-byte operand, as shown in Table 1-7. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
SHUFPD <i>xmm1, xmm2/mem128, imm8</i>	66 0F C6 /r ib	Shuffles packed double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and puts the result in the destination XMM register.

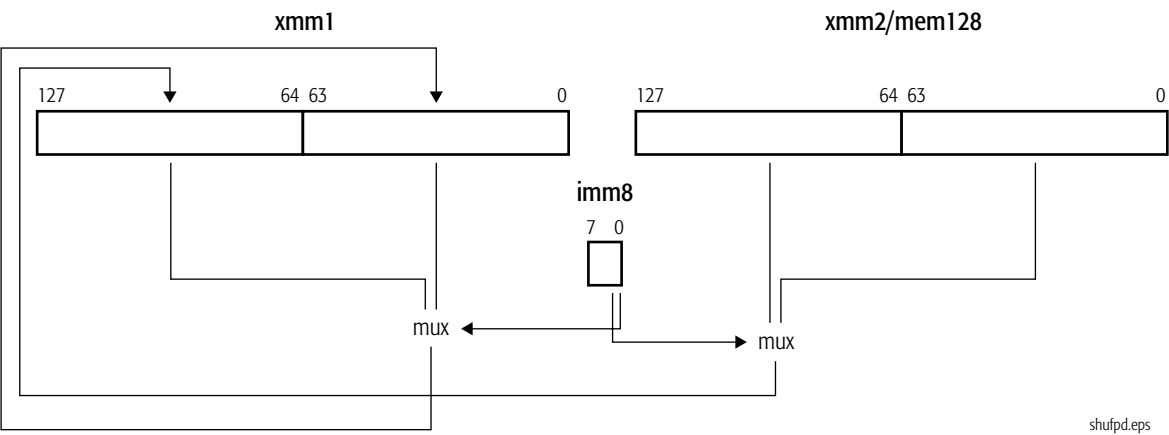


Table 1-7. Immediate-Byte Operand Encoding for SHUFPD

Immediate-Byte Bit Field	Value of Bit Field	Destination Bits Filled	Source 1 Bits Moved	Source 2 Bits Moved
0	0	63–0	63–0	—
	1	63–0	127–64	—
1	0	127–64	—	63–0
	1	127–64	—	127–64

Related Instructions

SHUFPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.

Exception	Real	Virtual 8086	Protected	Cause of Exception
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

SHUFPS**Shuffle Packed Single-Precision Floating-Point**

The SHUFPS instruction moves two of the four packed single-precision floating-point values in the first source operand to the low-order quadword of the destination (first source) and moves two of the four packed single-precision floating-point values in the second source operand to the high-order quadword of the destination. In each case, the value of the destination doubleword is determined by a two-bit field in the immediate-byte operand, as shown in Table 1-8. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic**Opcode****Description**SHUFPS *xmm1, xmm2/mem128, imm8*OF C6/*r ib*

Shuffles packed single-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and puts the result in the destination XMM register.

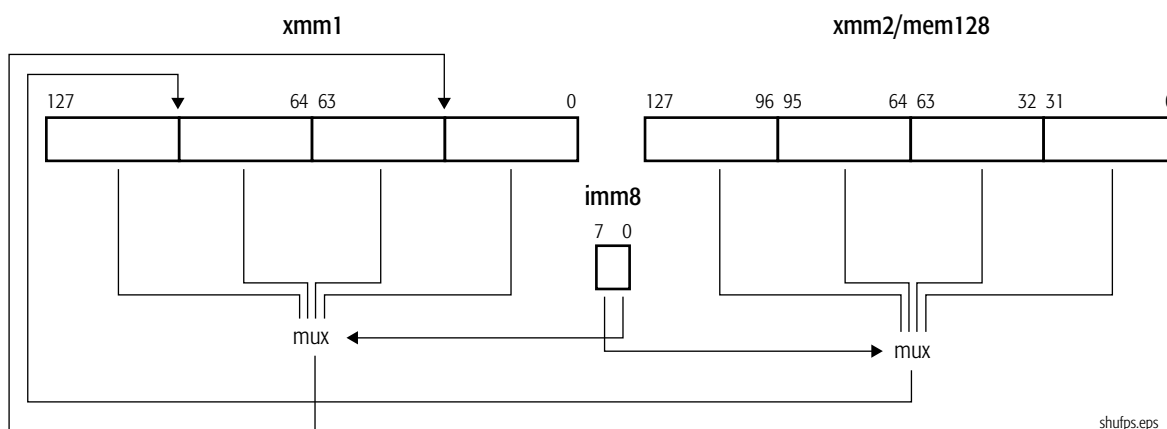


Table 1-8. Immediate-Byte Operand Encoding for SHUFPS

Immediate-Byte Bit Field	Value of Bit Field	Destination Bits Filled	Source 1 Bits Moved	Source 2 Bits Moved
1–0	0	31–0	31–0	—
	1	31–0	63–32	—
	2	31–0	95–64	—
	3	31–0	127–96	—
3–2	0	63–32	31–0	—
	1	63–32	63–32	—
	2	63–32	95–64	—
	3	63–32	127–96	—
5–4	0	95–64	—	31–0
	1	95–64	—	63–32
	2	95–64	—	95–64
	3	95–64	—	127–96
7–6	0	127–96	—	31–0
	1	127–96	—	63–32
	2	127–96	—	95–64
	3	127–96	—	127–96

Related Instructions

SHUFPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

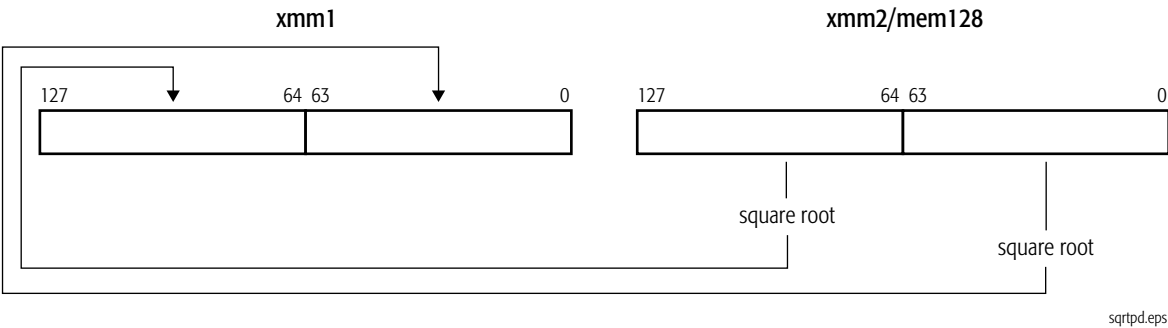
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

SQRTPD

Square Root Packed Double-Precision Floating-Point

The SQRTPD instruction computes the square root of each of the two packed double-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the corresponding quadword of another XMM register.

Mnemonic	Opcode	Description
SQRTPD <i>xmm1, xmm2/mem128</i>	66 0F 51 /r	Computes square roots of packed double-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

RSQRTPS, RSQRTSS, SQRTPS, SQRTSD, SQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M				M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Note: A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.																	

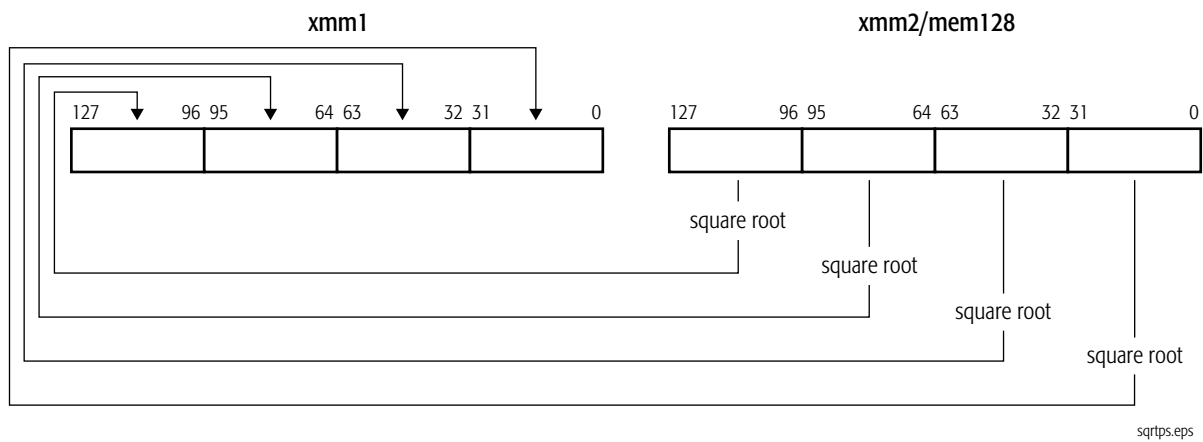
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value or is less than 0 (not including -0).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

SQRTPS**Square Root Packed Single-Precision Floating-Point**

The SQRTPS instruction computes the square root of each of the four packed single-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the corresponding doubleword of another XMM register.

Mnemonic	Opcode	Description
SQRTPS <i>xmm1, xmm2/mem128</i>	0F 51 /r	Computes square roots of packed single-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the destination XMM register.

**Related Instructions**

RSQRTPS, RSQRTSS, SQRTPD, SQRTSD, SQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M				M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

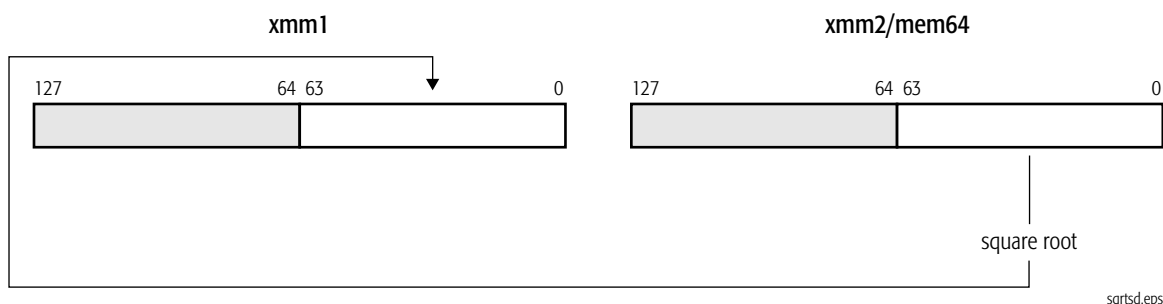
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value or is less than 0 (not including -0).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

SQRTSD**Square Root Scalar Double-Precision Floating-Point**

The SQRTSD instruction computes the square root of the low-order double-precision floating-point value in an XMM register or in a 64-bit memory location and writes the result in the low-order quadword of another XMM register. The high-order quadword of the destination XMM register is not modified.

Mnemonic	Opcode	Description
SQRTSD <i>xmm1, xmm2/mem64</i>	F2 0F 51 /r	Computes square root of double-precision floating-point value in an XMM register or 64-bit memory location and writes the result in the destination XMM register.

**Related Instructions**

RSQRTPS, RSQRTSS, SQRTPD, SQRTPS, SQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M				M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:

A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

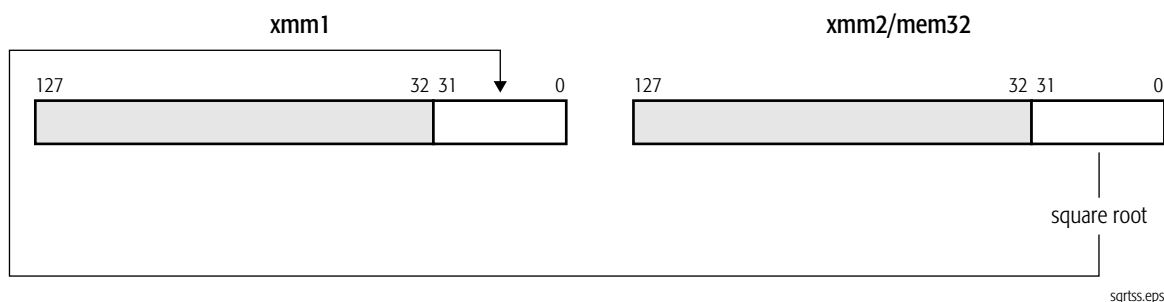
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value or is less than 0 (not including -0).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

SQRTSS Square Root Scalar Single-Precision Floating-Point

The SQRTSS instruction computes the square root of the low-order single-precision floating-point value in an XMM register or 32-bit memory location and writes the result in the low-order doubleword of another XMM register. The three high-order doublewords of the destination XMM register are not modified.

Mnemonic	Opcode	Description
SQRTSS <i>xmm1, xmm2/mem32</i>	F3 0F 51 /r	Computes square root of single-precision floating-point value in an XMM register or 32-bit memory location and writes the result in the destination XMM register.



Related Instructions

RSQRTPS, RSQRTSS, SQRTPD, SQRTPS, SQRTSD

rFLAGS Affected

None

MXCSR Flags Affected

	FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
												M				M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Note: A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.																	

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	The source operand is an SNaN value or is less than 0 (not including -0).
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

STMXCSR**Store MXCSR Control/Status Register**

The STMXCSR instruction saves the contents of the MXCSR register in a 32-bit location in memory. The MXCSR register is described in “Registers” in Volume 1.

Mnemonic	Opcode	Description
STMXCSR <i>mem32</i>	OF AE /3	Stores contents of MXCSR in 32-bit memory location.

Related Instructions

LDMXCSR

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

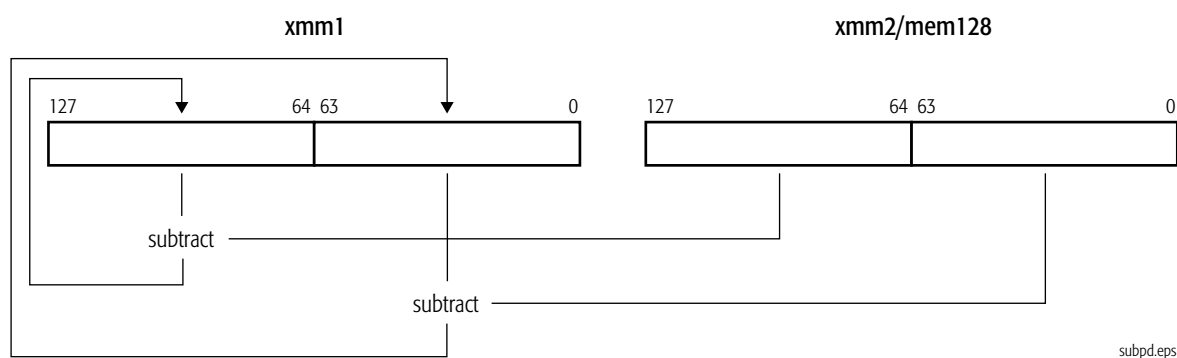
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).

Exception	Real	Virtual 8086	Protected	Cause of Exception
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)

SUBPD**Subtract Packed Double-Precision Floating-Point**

The SUBPD instruction subtracts each packed double-precision floating-point value in the second source operand from the corresponding packed double-precision floating-point value in the first source operand and writes the result of each subtraction in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
SUBPD <i>xmm1, xmm2/mem128</i>	66 0F 5C /r	Subtracts packed double-precision floating-point values in an XMM register or 128-bit memory location from packed double-precision floating-point values in another XMM register and writes the result in the destination XMM register.

**Related Instructions**

SUBPS, SUBSD, SUBSS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M		M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

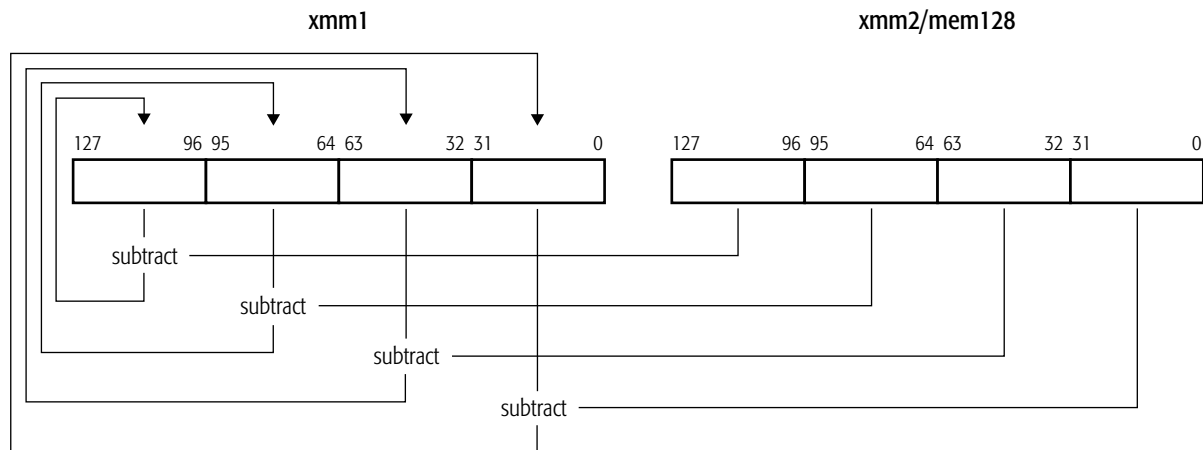
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand is an SNaN value or both source operands are infinities of the same sign.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

SUBPS**Subtract Packed Single-Precision Floating-Point**

The SUBPS instruction subtracts each packed single-precision floating-point value in the second source operand from the corresponding packed single-precision floating-point value in the first source operand and writes the result of each subtraction in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
SUBPS <i>xmm1, xmm2/mem128</i>	0F 5C/r	Subtracts packed single-precision floating-point values in an XMM register or 128-bit memory location from packed single-precision floating-point values in another XMM register and writes the result in the destination XMM register.

**Related Instructions**

SUBPD, SUBSD, SUBSS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M		M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

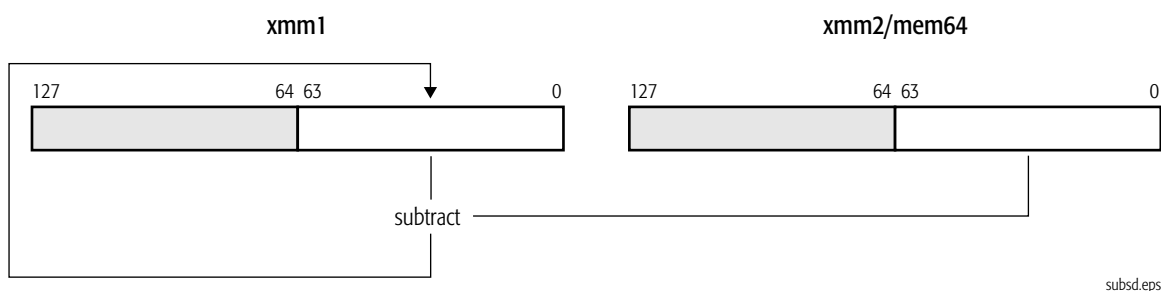
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand is an SNaN value or both source operands are infinities of the same sign.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

SUBSD Subtract Scalar Double-Precision Floating-Point

The SUBSD instruction subtracts the double-precision floating-point value in the low-order quadword of the second source operand from the double-precision floating-point value in the low-order quadword of the first source operand and writes the result in the low-order quadword of the destination (first source). The high-order quadword of the destination is not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 64-bit memory location.

Mnemonic	Opcode	Description
SUBSD <i>xmm1, xmm2/mem64</i>	F2 0F 5C/ <i>r</i>	Subtracts low-order double-precision floating-point value in an XMM register or in a 64-bit memory location from low-order double-precision floating-point value in another XMM register and writes the result in the destination XMM register.



Related Instructions

SUBPD, SUBPS, SUBSS

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M		M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

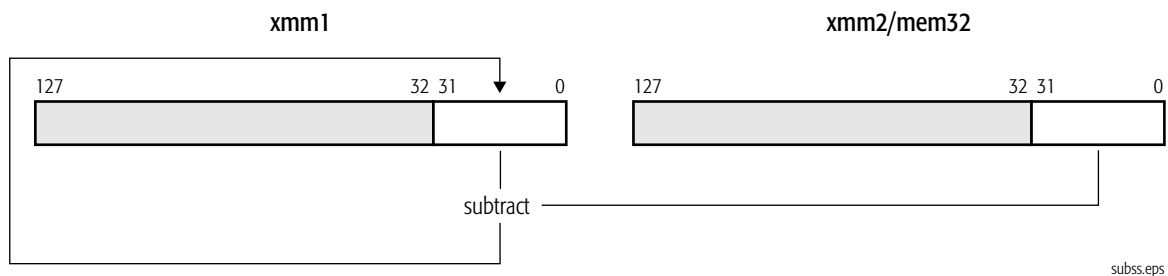
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand is an SNaN value or both source operands are infinities of the same sign.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

SUBSS**Subtract Scalar Single-Precision Floating-Point**

The SUBSS instruction subtracts the single-precision floating-point value in the low-order doubleword of the second source operand from the single-precision floating-point value in the low-order doubleword of the first source operand and writes the result in the low-order doubleword of the destination (first source). The three high-order doublewords of the destination are not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 32-bit memory location.

Mnemonic	Opcode	Description
SUBSS <i>xmm1</i> , <i>xmm2/mem32</i>	F3 0F 5C /r	Subtracts low-order single-precision floating-point value in an XMM register or in a 32-bit memory location from low-order single-precision floating-point value in another XMM register and writes the result in the destination XMM register.

**Related Instructions**

SUBPD, SUBPS, SUBSD

rFLAGS Affected

None

MXCSR Flags Affected

		FZ	RC			PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
													M	M	M		M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Note:
A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank. Reserved fields are shaded.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand is an SNaN value or both source operands are infinities of the same sign.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.
Overflow exception (OE)	X	X	X	The rounded result is too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	The rounded result is too small to fit into the format of the destination operand (either tiny, or if masked, tiny and inexact).
Precision exception (PE)	X	X	X	The result cannot be represented exactly in the destination format.

UCOMISD**Unordered Compare Scalar
Double-Precision Floating-Point**

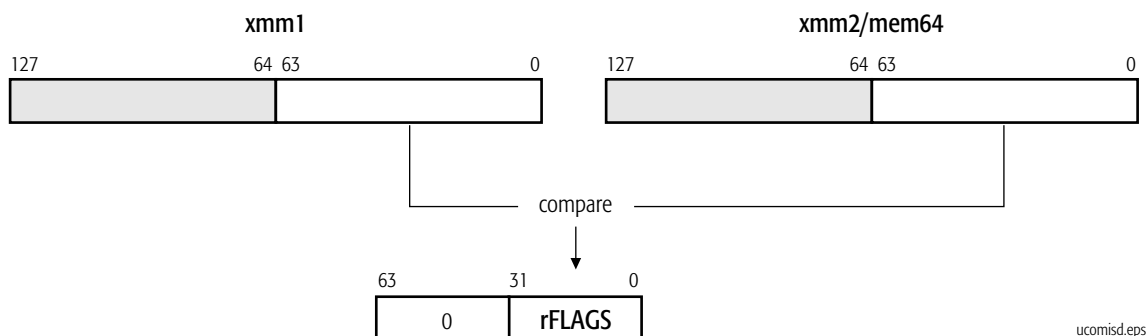
The UCOMISD instruction performs an unordered compare of the double-precision floating-point value in the low-order 64 bits of an XMM register with the double-precision floating-point value in the low-order 64 bits of another XMM register or a 64-bit memory location and sets the ZF, PF, and CF bits in the rFLAGS register to reflect the result of the compare. The result is unordered if one or both of the operand values is a NaN. The OF, AF, and SF bits in rFLAGS are set to zero.

If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

Mnemonic**Opcode****Description**UCOMISD *xmm1, xmm2/mem64*

66 0F 2E/r

Compares scalar double-precision floating-point values in an XMM register and an XMM register or 64-bit memory location. Sets rFLAGS.



ucomisd.eps

Result of Compare	ZF	PF	CF
Unordered	1	1	1
Greater Than	0	0	0
Less Than	0	0	1
Equal	1	0	0

Related Instructions

CMPPD, CMPPS, CMPSD, CMPSS, COMISD, COMISS, UCOMISS

rFLAGS Affected

ID	VIP	VIF	AC	VM	RF	NT	IOPL	OF	DF	IF	TF	SF	ZF	AF	PF	CF
								0				0	M	0	M	M
21	20	19	18	17	16	14	13–12	11	10	9	8	7	6	4	2	0

Note:
 Bits 31–22, 15, 5, 3, and 1 are reserved. A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank.
 If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

MXCSR Flags Affected

		FZ		RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																	M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Note: A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.																		

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)

Exception	Real	Virtual 8086	Protected	Cause of Exception
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand is an SNaN value or unsupported format.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

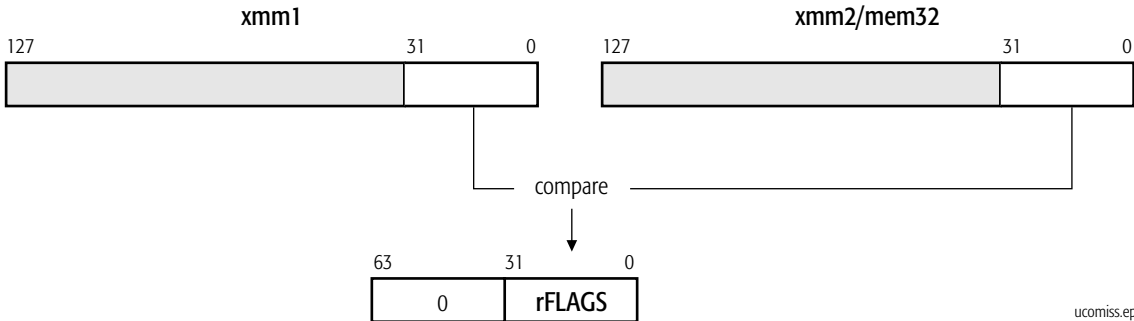
UCOMISS

Unordered Compare Scalar
Single-Precision Floating-Point

The UCOMISS instruction performs an unordered compare of the single-precision floating-point value in the low-order 32 bits of an XMM register with the single-precision floating-point value in the low-order 32 bits of another XMM register or a 32-bit memory location and sets the ZF, PF, and CF bits in the rFLAGS register to reflect the result. The result is unordered if one or both of the operand values is a NaN. The OF, AF, and SF bits in rFLAGS are set to zero.

If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

Mnemonic	Opcode	Description
UCOMISS <i>xmm1, xmm2/mem32</i>	0F 2E /r	Compares scalar single-precision floating-point values in an XMM register and an XMM register or 32-bit memory location. Sets rFLAGS.



Result of Compare	ZF	PF	CF
Unordered	1	1	1
Greater Than	0	0	0
Less Than	0	0	1
Equal	1	0	0

Related Instructions

CMPPD, CMPPS, CMPSD, CMPSS, COMISD, COMISS, UCOMISD

rFLAGS Affected

ID	VIP	VIF	AC	VM	RF	NT	IOPL	OF	DF	IF	TF	SF	ZF	AF	PF	CF
								0				0	M	0	M	M
21	20	19	18	17	16	14	13–12	11	10	9	8	7	6	4	2	0

Note:
 Bits 31–22, 15, 5, 3, and 1 are reserved. A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank.
 If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

MXCSR Flags Affected

		FZ	RC		PM	UM	OM	ZM	DM	IM		PE	UE	OE	ZE	DE	IE
																M	M
31	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Note: A flag that can be set to one or zero is M (modified). Unaffected flags are blank. Shaded fields are reserved.																	

Exceptions

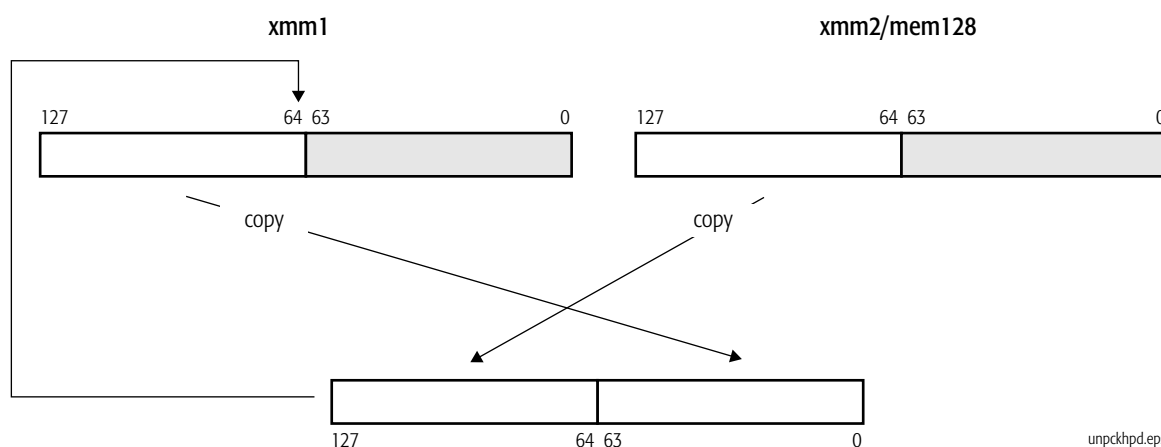
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is cleared to 0, and there is an unmasked SIMD floating-point exception. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).

Exception	Real	Virtual 8086	Protected	Cause of Exception
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference resulted from the instruction execution, and the alignment check bit (AC) of the rFLAGS register and the alignment mask bit (AM) of CR0 are set to 1. (In protected mode, CPL = 3.)
SIMD Floating-Point Exception, #XF	X	X	X	The operating-system unmasked-exception support bit (OSXMMEXCPT) of CR4 is set to 1, and there is an unmasked SIMD floating-point exception. See <i>SIMD Floating-Point Exceptions</i> , below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand is an SNaN value or unsupported format.
Denormalized-operand exception (DE)	X	X	X	A source operand is a denormal value.

UNPCKHPD**Unpack High Double-Precision Floating-Point**

The UNPCKHPD instruction unpacks the high-order double-precision floating-point values in the first and second source operands and packs them into quadwords in the destination (first source). The value from the first source operand is packed into the low-order quadword of the destination, and the value from the second source operand is packed into the high-order quadword of the destination. The low-order quadwords of the source operands are ignored. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
UNPCKHPD <i>xmm1, xmm2/mem128</i>	66 0F 15 /r	Unpacks high-order double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and packs them into the destination XMM register.



unpckhpd.eps

Related Instructions

UNPCKHPS, UNPCKLPD, UNPCKLPS

rFLAGS Affected

None

MXCSR Flags Affected

None

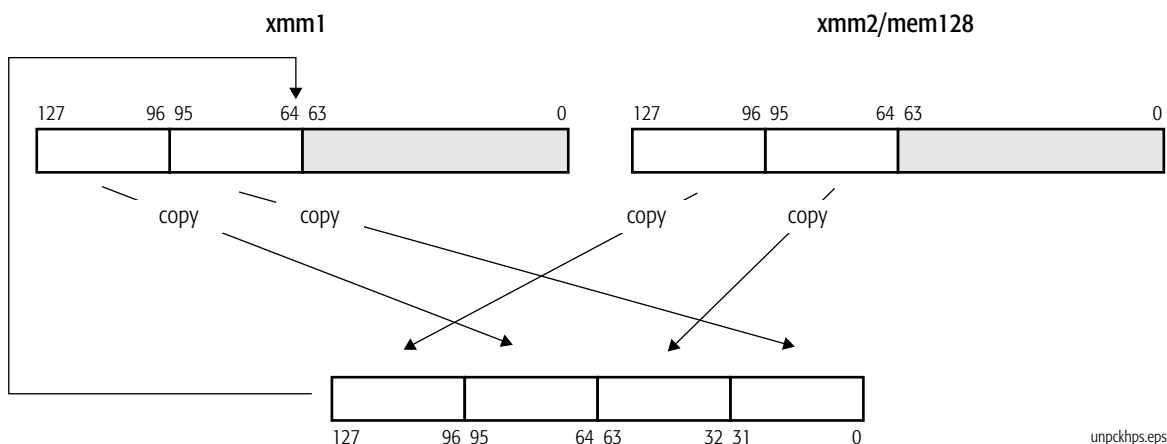
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

UNPCKHPS**Unpack High Single-Precision Floating-Point**

The UNPCKHPS instruction unpacks the high-order single-precision floating-point values in the first and second source operands and packs them into interleaved doublewords in the destination (first source). The low-order quadwords of the source operands are ignored. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
UNPCKHPS <i>xmm1, xmm2/mem128</i>	OF 15/ <i>r</i>	Unpacks high-order single-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and packs them into the destination XMM register.



unpckhps.eps

Related Instructions

UNPCKHPD, UNPCKLPD, UNPCKLPS

rFLAGS Affected

None

MXCSR Flags Affected

None

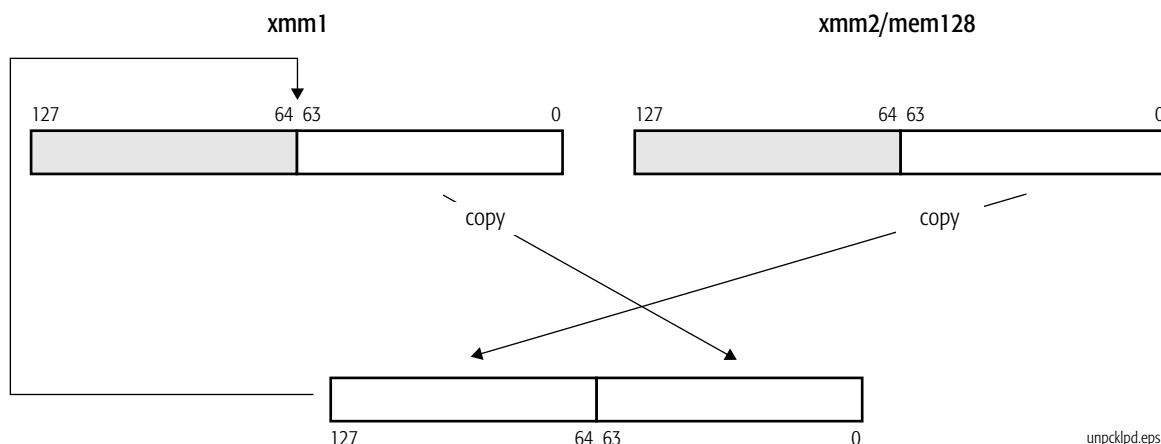
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

UNPCKLPD**Unpack Low Double-Precision Floating-Point**

The UNPCKLPD instruction unpacks the low-order double-precision floating-point values in the first and second source operands and packs them into the destination (first source). The value from the first source operand is packed into the low-order quadword of the destination, and the value from the second source operand is packed into the high-order quadword of the destination. The high-order quadwords of the source operands are ignored. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
UNPCKLPD <i>xmm1, xmm2/mem128</i>	66 0F 14 /r	Unpacks low-order double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and packs them into the destination XMM register.

**Related Instructions**

UNPCKHPD, UNPCKHPS, UNPCKLPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

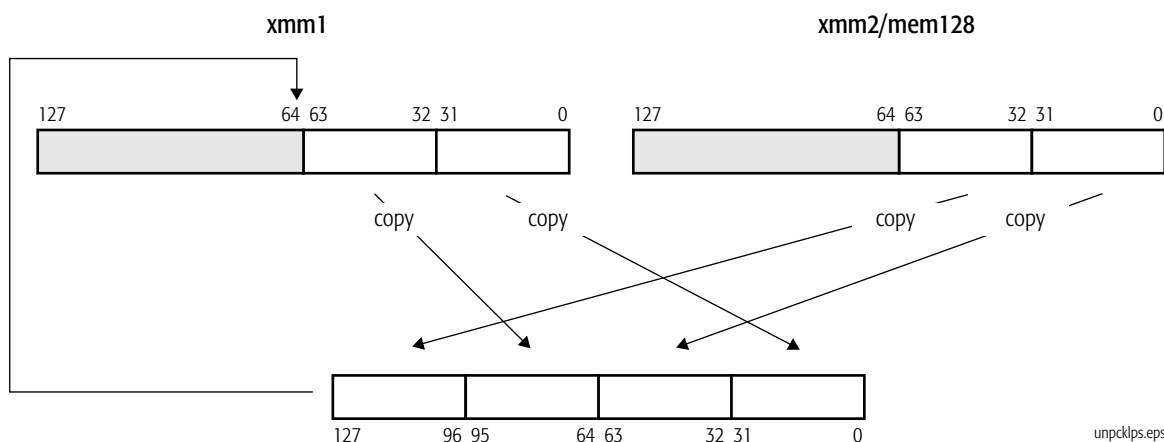
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

UNPCKLPS

Unpack Low Single-Precision Floating-Point

The UNPCKLPS instruction unpacks the low-order single-precision floating-point values in the first and second source operands and packs them into interleaved doublewords in the destination (first source). The high-order quadwords of the source operands are ignored. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location

Mnemonic	Opcode	Description
UNPCKLPS <i>xmm1, xmm2/mem128</i>	0F 14/r	Unpacks low-order single-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and packs them into the destination XMM register.



Related Instructions

UNPCKHPD, UNPCKHPS, UNPCKLPD

rFLAGS Affected

None

MXCSR Flags Affected

None

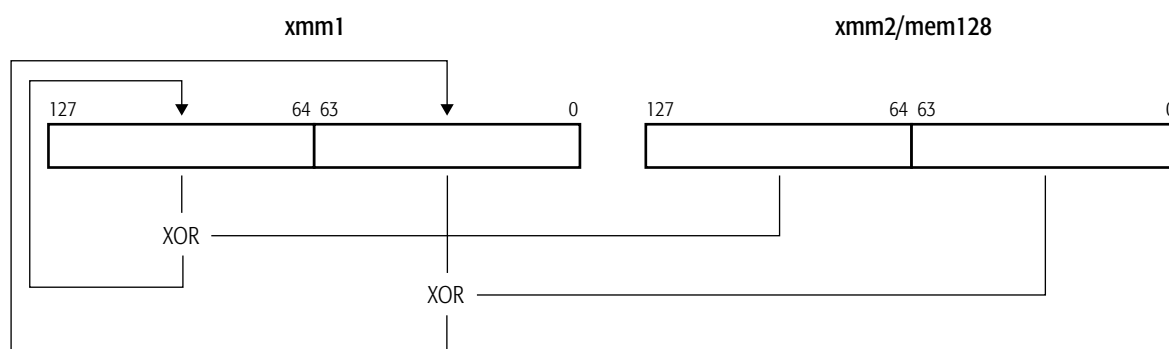
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

XORPD**Logical Bitwise Exclusive OR
Packed Double-Precision Floating-Point**

The XORPD instruction performs a bitwise logical Exclusive OR of the two packed double-precision floating-point values in the first source operand and the corresponding two packed double-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
XORPD <i>xmm1, xmm2/mem128</i>	66 0F 57 /r	Performs bitwise logical XOR of two packed double-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



xorpd.eps

Related Instructions

ANDNPD, ANDNPS, ANDPD, ANDPS, ORPD, ORPS, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

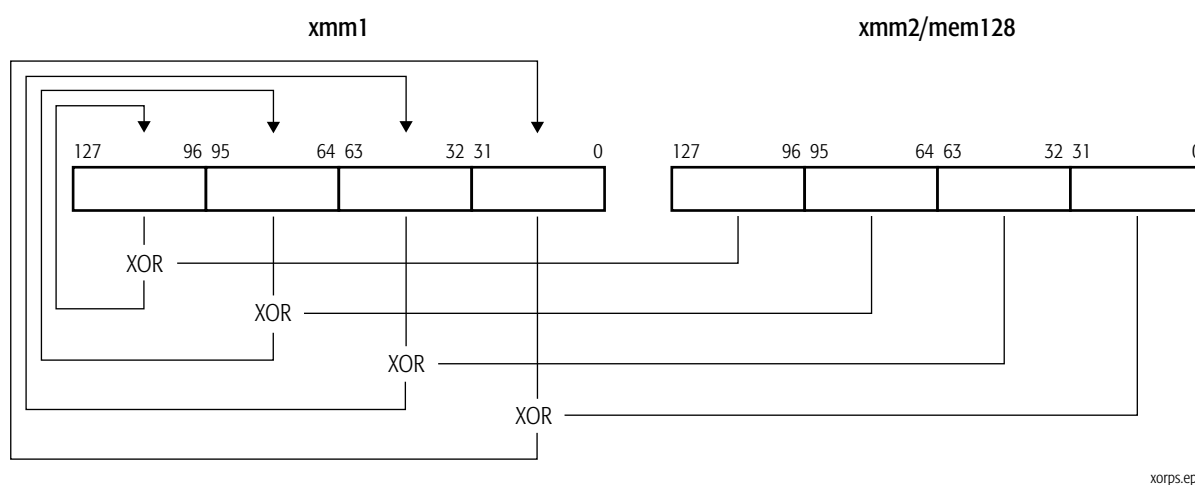
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE2 bit (bit 26) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

XORPS**Logical Bitwise Exclusive OR
Packed Single-Precision Floating-Point**

The XORPS instruction performs a bitwise Exclusive OR of the four packed single-precision floating-point values in the first source operand and the corresponding four packed single-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Mnemonic	Opcode	Description
XORPS <i>xmm1, xmm2/mem128</i>	0F 57 /r	Performs bitwise logical XOR of four packed single-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



xorps.eps

Related Instructions

ANDNPD, ANDNPS, ANDPD, ANDPS, ORPD, ORPS, XORPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The emulate bit (EM) of CR0 is set to 1. The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0. The SSE bit (bit 25) in CPUID extended function 8000_0001 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 is set to 1.
Stack, #SS			X	During instruction execution, the effective address of a stack operand points to an illegal stack-memory location. (Illegal references to non-stack memory are reported by #GP.)
General protection, #GP	X	X	X	The memory operand is not aligned on a 16-byte boundary.
General protection, #GP			X	During instruction execution, the effective address of one of the operands points to an illegal memory location (except that illegal stack-segment memory references are reported by #SS).
General protection, segment overrun, #GP	X	X		The address of a data operand is outside the address range 0000h to FFFFh.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

Index

Numerics

16-bit mode.....	xii
32-bit mode.....	xiii
64-bit mode.....	xiii

A

ADDDPD	4
ADDPS	6
addressing, RIP-relative.....	xviii
ADDSD	8
ADDSS	11
ANDNPD	14
ANDNPS	16
ANDPD	18
ANDPS	20

B

biased exponent.....	xiii
----------------------	------

C

CMPPD	22
CMPPS	26
CMPSD	29
CMPSS	32
COMISD.....	35
COMISS	38
commit	xiii
compatibility mode.....	xiii
CVTDQ2PD	41
CVTDQ2PS	43
CVTPD2DQ	45
CVTPD2PI	47
CVTPD2PS	50
CVTPI2PD	53
CVTPI2PS.....	55
CVTPS2DQ	58
CVTPS2PD	60
CVTPS2PI.....	62
CVTSD2SI	65
CVTSD2SS.....	68
CVTSI2SD	71
CVTSI2SS	74
CVTSS2SD.....	77
CVTSS2SI	80
CVTTPD2DQ	83
CVTTPD2PI.....	85
CVTTPS2DQ	88
CVTTPS2PI	90
CVTTSD2SI	93

CVTTSS2SI	96
-----------------	----

D

direct referencing.....	xiv
displacements.....	xiv
DIVPD	99
DIVPS	102
DIVSD	105
DIVSS	108
double quadword.....	xiv
doubleword	xiv

E

eAX–eSP register	xx
effective address size.....	xiv
effective operand size.....	xv
eFLAGS register.....	xx
eIP register	xxi
element	xv
endian order	xxiii
exception.....	xv
exponent	xiii

F

flush.....	xv
FXRSTOR	111
FXSAVE	113

I

IGN	xv
indirect.....	xv
instructions	
128-bit media	1
SSE	1
SSE-2	1

L

LDMXCSR	115
legacy mode.....	xvi
legacy x86	xvi
long mode.....	xvi
LSB	xvi
lsb	xvi

M

mask	xvi
MASKMOVDQU	117
MAXPD	119
MAXPS	121
MAXSD	124
MAXSS	127

MBZ.....	xvii	overflow.....	xvii
MINPD.....	130	P	
MINPS.....	132	packed.....	xvii
MINSD.....	135	PACKSSDW.....	207
MINSS.....	138	PACKSSWB.....	209
modes		PACKUSWB.....	211
16-bit.....	xii	PADDB.....	213
32-bit.....	xiii	PADDD.....	215
64-bit.....	xiii	PADDQ.....	217
compatibility.....	xiii	PADDSB.....	219
legacy.....	xvi	PADDSW.....	221
long.....	xvi	PADDUSB.....	223
protected.....	xviii	PADDUSW.....	225
real.....	xviii	PADDW.....	227
virtual-8086.....	xx	PAND.....	229
moffset.....	xvii	PANDN.....	231
MOVAPD.....	141	PAVGB.....	233
MOVAPS.....	143	PAVGW.....	235
MOVD.....	146	PCMPEQB.....	237
MOVDQ2Q.....	149	PCMPEQD.....	239
MOVDQA.....	151	PCMPEQW.....	241
MOVDQU.....	153	PCMPGTB.....	243
MOVHLPS.....	155	PCMPGTD.....	245
MOVHPD.....	157	PCMPGTW.....	247
MOVHPS.....	159	PEXTRW.....	249
MOVLHPS.....	161	PINSRW.....	251
MOVLPD.....	163	PMADDWD.....	254
MOVLPS.....	165	PMAXSW.....	256
MOVMSKPD.....	167	PMAXUB.....	258
MOVMSKPS.....	169	PMINSW.....	260
MOVNTDQ.....	171	PMINUB.....	262
MOVNTPD.....	173	PMOVMskb.....	264
MOVNTPS.....	175	PMULHUW.....	266
MOVQ.....	177	PMULHW.....	268
MOVQ2DQ.....	179	PMULLW.....	270
MOVSD.....	181	PMULUDQ.....	272
MOVSS.....	184	POR.....	274
MOVUPD.....	187	protected mode.....	xviii
MOVUPS.....	189	PSADBW.....	276
MSB.....	xvii	PSHUF.....	278
msb.....	xvii	PSHUFHW.....	281
MSR.....	xxi	PSHUFLW.....	284
MULPD.....	191	PSLLD.....	287
MULPS.....	194	PSLLDQ.....	289
MULSD.....	197	PSLLQ.....	291
MULSS.....	200	PSLLW.....	293
O		PSRAD.....	295
octword.....	xvii	PSRAW.....	298
offset.....	xvii	PSRLD.....	301
ORPD.....	203	PSRLDQ.....	303
ORPS.....	205	PSRLQ.....	305

PSRLW	307	SSE	xix
PSUBB	310	SSE-2	xix
PSUBD	312	sticky bit.....	xix
PSUBQ	314	STMXCSR.....	367
PSUBSB	316	SUBPD.....	369
PSUBSW	318	SUBPS	372
PSUBUSB	320	SUBSD.....	375
PSUBUSW	322	SUBSS	378
PSUBW	324	T	
PUNPCKHBW	326	TSS.....	xix
PUNPCKHDQ	328	U	
PUNPCKHQDQ	330	UCOMISD	381
PUNPCKHWD	332	UCOMISS.....	384
PUNPCKLBW	334	underflow.....	xix
PUNPCKLDQ	336	UNPCKHPD.....	387
PUNPCKLQDQ	338	UNPCKHPS	389
PUNPCKLWD	340	UNPCKLPD	391
PXOR	342	UNPCKLPS.....	393
Q		V	
quadword.....	xviii	vector.....	xix
R		virtual-8086 mode	xx
r8–r15.....	xxi	X	
rAX–rSP.....	xxi	XORPD.....	395
RAZ.....	xviii	XORPS	397
RCPPS	344		
RCPSS.....	346		
real address mode. See real mode			
real mode.....	xviii		
registers			
eAX–eSP	xx		
eFLAGS.....	xx		
eIP	xxi		
r8–r15.....	xxi		
rAX–rSP	xxi		
rFLAGS	xxii		
rIP.....	xxii		
relative.....	xviii		
rFLAGS register.....	xxii		
rIP register	xxii		
RIP-relative addressing.....	xviii		
RSQRTPS	348		
RSQRTSS.....	350		
S			
set	xviii		
SHUFPD	352		
SHUFPS.....	355		
SQRTPD.....	358		
SQRTPS	360		
SQRSD	363		
SQRTSS	365		

